VŠB – Technická Univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Archivace a externí zpřístupnění souborového systému virtualizovaných stanic systému Virtlab

Archivation of Filesystem of Virtualized PCs of Virtlab system and External Access to Archived Files

2012 Roman Bližňák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Roman Bližňák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Archivace a externí zpřístupnění souborového systému virtualizovaných stanic systému Virtlab
Archivation of Filesystem of Virtualized PCs of Virtlab System and External Access to Archived Files

Zásady pro vypracování:

Cílem práce je poskytnout možnost uchovávání obsahu souborového systému uživatelských PC realizovaných v systému Virtlab a jeho převodu mezi jednotlivými rezervovanými úlohami. Smyslem je poskytnout uživatelům možnost přerušované práce na témže problému při zachování konfigurace.

1. Seznamte se s architekturou systému Virtlab a způsobem simulace uživatelských PC.
2. Navrhněte strukturu archivu záloh souborových systému a přístupová práva k nim s ohledem na možnost skupinové spolupráce více uživatelů na společné rezervaci.
3. Navrhněte způsob uschování souborového systému jednotlivých PC a jejich následné obnovení.
4. Navrhněte a do systému integrujte uživatelské rozhraní pro specifikaci požadavku převodu obsahu souborového systému mezi rezervacemi téže úlohy a pro okamžité uložení, resp. obnovení obsahu souborového systému do/z archivu na žádost uživatele.
5. Zajistěte možnost uploadu/downloadu souboru do/z archivu z veřejného Internetu.

Seznam doporučené odborné literatury:

JUŠKA, Pavel. Virtualizační platforma pro dynamickou aktivaci specifických virtuálních instancí s OS Linux. Ostrava, 2010. 49 s. Diplomová práce. VŠB-TU Ostrava, FEI.
Stránky projektu Virtlab [online]. 2006-06-28 [cit. 2010-11-05]. Virtuální laboratoř počítačových sítí. Dostupné z WWW: <http://www.virtlab.cz>.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Kateřina Bambušková**

Datum zadání: 19.11.2010
Datum odevzdání: 04.05.2012

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 19. 12. 2011 ***Podpis***

Abstrakt

Tato diplomová práce navazuje na školní projekt Virtlab (Virtuální laboratoř počítačových sítí), což je prostředí umožňující zpřístupnit laboratorní prvky a následně z nich vytvořit počítačovou síť, se kterou mohou studenti vzdáleně pracovat a konfigurovat ji, jako by to byla reálná počítačová síť. Cílem této práce je systém Virtlab rozšířit o funkcionalitu trvalé archivace souborového systému virtuálních počítačů realizovaných v systému Virtlab. Student bude mít možnost s takto archivovanými systémy pracovat napříč jednotlivými rezervovanými úlohami a také tyto systémy spravovat. U řešení je kladen požadavek na optimální využití diskového prostoru. Smyslem práce je poskytnout uživatelům možnost přerušované práce na témže problému při zachování konfigurace.

Klíčová slova

virtuální stroj, virtuální PC, správa virtuálních strojů, snapshot, záloha, Virtlab

Abstract

This dissertation extends the school project Virtlab (Virtual Networking Laboratory) which is an environment capable to make accessible laboratory items and subsequently from these items create a network with which students can work and configure it like it was a real computer network. The aim of this dissertation is to extend Virtlab with the functionality persistent archiving of a file system of virtual computers realized in the system Virtlab. Student will be allowed to work with those archived systems across particular reserved tasks and manage those systems as well. The solution requires optimal utilization of the disk space. The sense of the dissertation is to provide a possibility to users to work intermittently with the same issue and sustain the same configuration.

Keywords

virtual machine, virtual PC, management of virtua machines, snapshot, backup, Virtlab

Seznam použitých zkratek

Virtlab - Virtual Networking Laboratory
VM - Virtual Machine
PC - Personal Computer
OS - Operating System
HW - Hardware
SW - Software
VMM - Virtual Machine Monitor (hypervisor)
HVM - Hardware Virtual Machine
KVM - Kernel-based Virtual Machine
VLAN - Virtual Local Area Network
SSH - Secure Shell
LVM - Logical Volume Management
PV - Physical Volume
VG - Group Volume
LV - Logical Volume
IP - Internet Protocol
HTML - HyperText Markup Language
HTTP - HyperText Transfer Protocol
PHP - Hypertext Preprocesor
STD - State Transition Diagram

Obsah

Seznam obrázků

1 Úvod

V době, kdy svět řeší globální krizi a firmy jsou pod rostoucím tlakem nuceny omezovat výdaje všude, kde je to jen trochu možné, si ve světě IT získává stále větší popularitu řešení, které ve firmách pomáhá zoptimalizovat firemní IT segment a tím i efektivně snížit náklady, na což firmy v době krize velice rády slyší. Řeč je o virtualizaci, která vhodnou implementací vede ke zlepšení využití stávajích zdrojů, snížení nákladů, zvýšení produktivity podnikání a dalších výhod plnoucích z používání této technologie. Je velice pravděpodobné, že tento trend se v následujících letech ještě urychlí, nejen díky stále přetrvávající světové krizi, ale také neustálé rostoucí výkonnosti hardwaru, kdy jeho potenciál není častokrát zdaleka využit, ale virtualizace tento problém dokáže efektivně řešit.

V této diplomové práci je virtualizace použita ve školním systému Virtlab, který studentům slouží k procvičování a prohlubování znalostí počítačových sítí. Ve Virtlabu si lze rezervovat laboratorní síťové prvky, ze kterých studenti mohou následně vytvářet počítačové sítě a pracovat s nimi, jako by to byly skutečné sítě. Součástí takto vytvořené sítě nejsou pouze síťové prvky jako router a switch, ale i klasické koncové počítače, které jsou virtualizovány a na které je tato diplomová práce zaměřena.

1.1 Cíl práce

Cílem práce je navrhnout a implementovat systém archivace a správy virtuálních PC, které jsou ve Virtlabu zastoupeny, tak aby je student mohl v budoucnu kdykoliv využít u jiné úlohy a nemusel provádět konfiguraci na virtuálním stroji opět znova. Prvním krokem k realizaci diplomové práce je seznámit se s architekturou systému Virtlab a způsobem simulace uživatelských PC. Na základě prvního kroku navrhneme několik alternativ řešení z nichž zvolíme jedno, které bude nejlépe odpovídat požadavkům již existující architektury Virtlabu a toto řešení poté implementujeme.

Kapitola 2 – V druhé kapitole je blíže nastíněna problematika virtualizace a současné dostupné technologie pro virtualizaci s jejich typickými zástupci.

Kapitola 3 – Třetí kapitola se zaměřuje na analýzu požadavků pro archivaci virtuálních PC v systému Virtlab, návrh několika řešení a výběr jednoho řešení na základě analýzy.

Kapitola 4 – Ve čtvrté kapitole je podrobně popsána samotná implementace řešení pro archivaci virtuálních PC a jejich následném znovuužití.

Kapitola 5 – Pátá kapitola detailně popisuje provázání kompletního naimplementovaného řešení se systémem Virtlab.

2 Virtualizace

Virtualizace, slovo, které značně zdomácnělo v IT slovníku v posledních letech a stal se z něj fenomén současných informačních technologií. O žádné jiné technologii se v poslední době nemluví tolik jako o virtualizaci. Není pochyb, že velký podíl na tom sehrál značný pokrok virtualizace v posledních letech, který se ubíral doslova mílovým tempem a také nepřeberné množství řešení, které dnešní trh s virtualizací nabízí. Firmy, které virtualizaci nabízejí a implementují shodně tvrdí, že virtualizace podnikům přináší oproti klasickému řešení obrovské finanční úspory. Jaké jsou však další možnosti této technologie? Co vše tato technologie vyžaduje, aby přinesla kýžené výsledky? Má tato technologie i stinné stránky? Na všechny předešlé otázky se pokusí odpovědět následující kapitola.

Historie virtualizace

Koncept virtualizace byl poprvé představen firmou IBM v šedesátých letech za účelem plně utilizovat hardware mainframeových serverů. Hnacím motorem pro virtualizaci zde byla obrovská pořizovací cena těchto serverů. Virtualizace je založena na rozdělení mainframe hardwaru na logické části, které tvoří virtuální stroje. Toto rozdělení umožňovalo mainframu provádět multitasking a spouštět tak v jeden čas několik aplikací a procesů. První představitel této technologie byl mainframe CP-40z rodiny počítačů IBM-360. Komerčně úspěšný se pak stal následující model CP-67. Vývoj virtualizace pak dále pokračoval v 70. letech na nové platformě IBM System/370.

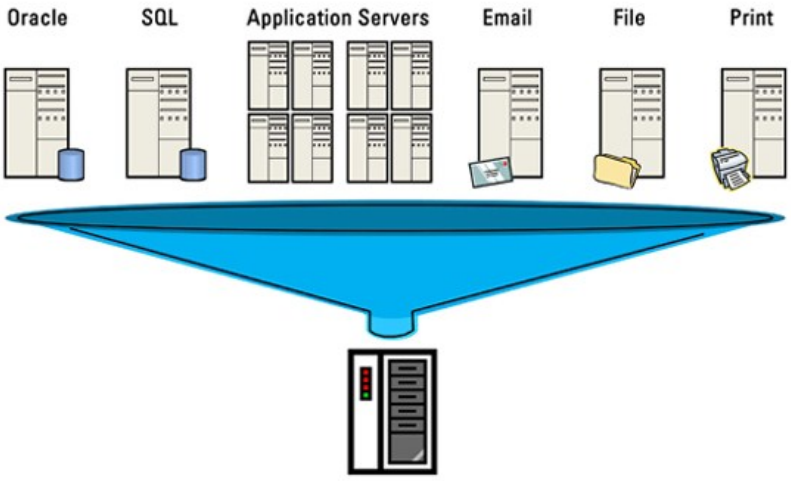
Přestože virtualizace v rámci OS/370 byla pro řadu zákazníků zajímavá, vyžadovala velmi rozsáhlou hardwarovou podporu, která zvyšovala cenu. Ostatní výrobci počítačů proto plnou virtualizaci nenabízeli a zájem o ni prakticky zmizel v souvislosti se zavedením osobních počítačů v 80. a 90. letech (ty totiž nabízely mnohem více fyzických počítačů, než byla tehdejší technologie schopná nabídnout počítačů virtuálních a to za mnohem lepších finančních a provozních podmínek). S růstem výkonu počítačů v 90. letech a jejich nasazením v podobě serverů využívajích stejné procesory i základní architekruru se však virtualizace znovu vrátila do hry a umožňuje tak naplnit jeden ze slibů informatiky - plnou individualizaci prostředí při vysoce efektivním využití zdrojů.

Pojem „virtualizace“

Pokud bychom měli ve zkratce definovat obecně pojem virtualizace, co znamená a co si pod tímto pojmem představit, tak zřejmě nejlépe tomuto požadavku odpovídá definice – *„Jako virtualizace se v IT prostředí označují postupy a techniky, které umožňují k dostupným zdrojům přistupovat jiným způsobem, než jakým fyzicky existují.“* [8]

Virtualizované prostředí může být mnohem snáze přizpůsobeno potřebám uživatelů, snáze se používat, případně před uživateli zakrývat pro ně nepodstatné detaily (jako např. rozmístění hardwarových prostředků). Virtualizovat lze na různých úrovních, od celého počítače (tzv. virtuální stroj), po jeho jednotlivé hardwarové komponenty (např. virtuální procesory, paměť atd.), případně pouze softwarové prostředí (virtualizace operačního systému).

Nejčastěji však bývá virtualizace vnímána, jako běh více virtuálních strojů na jednom fyzickém stroji. viz. Obrázek 2.1. Nemusí to však být pravda, jelikož existuje více typů virtualizací, o kterých se budu podrobněji zmiňovat v této kapitole.



Obrázek 2.1: Tradiční zobrazení virtualizace

2.1 Druhy virtualizace

Virtualizaci lze rozdělit do několika typů podle toho, co konkrétně virtualizujeme. Je možno virtualizovat od jednotlivých periférií počítače až po kompletní počítač, který se navenek chová jako plně autonomní stroj. Nejznámější a nejhojněji používaná je hardwarová virtualizace. Poměrně často je využívaná i softwarová virtualizace. Dále následuje výčet virtualizací, které virtualizují jednotlivé části počítače, jako virtualizace operačních pamětí, diskových úložišť, dat, virtualizace počítačové sítě, emulace jako speciální typ virtualizace apod. V dalších podkapitolách jsou podrobněji rozepsány nejvýznamější typy virtualizací.

2.1.1 Hardwarová virtualizace

Hardwarová virtalizace se odkazuje na virtuální stroj, který se chová jako nezávislý reálný počítač s operačním systémem. Software, který je spouštěn na tomto typu virtuálního stroje, je jasně oddělen od fyzické hardwarové vrstvy, tzn. že například na stroji s operačním systémem Windows, může běžet virtuální stroj s operačním systémem Linux, kde je možno bez jakýchkoliv omezení spustit software určený pro Linuxové distribuce.

Plná virtualizace

U tohoto typu virtualizujeme téměř všechny součásti počítače, pouze procesor se nevirtualizuje, tudíž fyzický i virtuální stroj musí používat shodnou platformu procesoru. V takovémto virtuálním prostředí, hostovaný operační systém nemůže žádným způsobem poznat, že nemá přístup k fyzickému technickému vybavení (hostovaný systém není nijak modifikován). Operační systém ani aplikační programy nepotřebují žádné modifikace. Jedná se v podstatě o ideální stav, kdy dochází k plnému oddělení fyzické vrstvy, veškeré programy běží pouze na virtuálním hardware a přístup k fyzickému vybavení je vždy zprostředkován.

Tento typ virtualizac má řadu výhod - můžeme virtuální prostředí navrhnout tak, aby nám vyhovovalo (velikost paměti, typ procesoru, typ a kapacitu disku apod.). Hostitelský ani hostovaný operační systém nemusí být nijak modifikován. Naopak největší slabinou jsou veškeré vstupní a výstupní operace, které jsou obtěžkány značnou režií – v případě, že se hostovaný operační systém pokusí přistoupit k fyzickému hardwaru, je takový požadavek odchycen a upraven (čtení z virtuálního disku je třeba převést na čtení určitého místa na disku fyzickém apod.) a následně předán ke zpracování hostitelskému systému. Výsledek putuje stejně strastiplným způsobem zpět do hostovaného systému.

Paravirtualizace

Jelikož u plné virtualizace dochází úplnému oddělení fyzické a virtualizační vrstvy, je zde prakticky nemožné dosáhnout plného výkonu a to i v případě, že virtuální počítač je přesným obrazem hardwaru, na kterém běží. Je to způsobeno tím, že většina počítačových periférií je virtualizačním softwarem emulována a většinu operací provádí virtualizační program sám, namísto aby je přímo vykonával hardware. Pokud však předpokládáme, že se alespoň některé komponenty virtuálního a fyzického počítače shodují, pak můžeme odstoupit od principu plné virtualizace a pracovat s tzv. paravirtualizací. Ta se vyznačuje tím, že provádí jen částečnou abstrakci na úrovni virtuálního počítače, tj. nabízí virtuální prostředí, které je podobné tomu fyzickému, na kterém virtuální počítač provozujeme. To umožňuje, aby virtuální počítač v maximální míře využíval vlastnosti fyzického prostředí (neemulují se komponenty virtuálního PC).

Nezbytný předpoklad pro paravirtualizaci je speciálně upravené jádro hostujícího operačního systému s tzv. hypervisorem, který poskytuje hostovaným systémům přístup k hardware. Hostované operační systémy jsou taktéž upraveny - ví, že nemají přístup k fyzickému hardware, takže všechny přístupy k hardware jsou převedeny na volání hypervisoru. Virtualizace v tomto případě není úplná a hostovaný operační systém může rozpoznat, že běží ve virtuálním prostředí. [9]

Virtualizace s podporou hardwaru

Jedná se o typ virtualizace, kde je nejužší vztah mezi virtualizovaným OS a vrstvou fyzického hardware. Intel i AMD poskytují u některých typů svých procesorů podporu pro tuto technologii - Intel VT-x a AMD-V. V současnosti se jedná o nejčastěji používané řešení, protože poskytuje vyšší výkon než pouhá hardwarová emulace (plná virtualizace) a zároveň umožňuje použití neupraveného softwaru. Jako nejlepší řešení se však ukazuje hybridní virtualizace, což je kombinace virtualizace s podporou hardware a výše zmíněné paravirtualizace. Využití hardwarové podpory virtualizace jsou schopny mnohé nástroje - XEN, Vmware ESX, VirtualBox. [10]

Emulátor

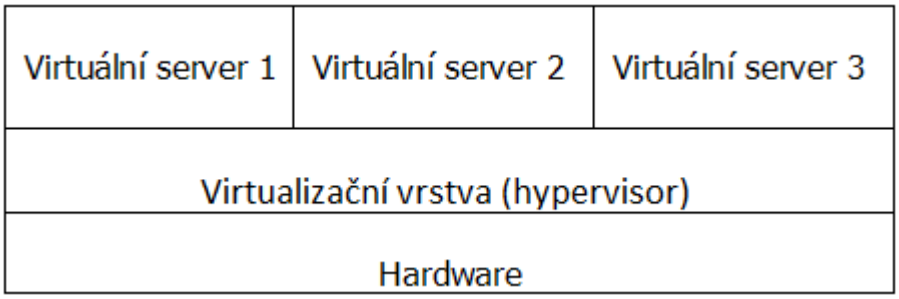
Emulátor je druh softwaru umožňující běh počítačových programů na jiné platformě (architektuře, operačním systému), než pro kterou byly původně vytvořeny a kterou samy od sebe podporují [11]. Základním principem emulátoru je překlad strojových instrukcí hostovaného systému na strojové instrukce hostitelského stroje. Jinými slovy emuluje se i procesor včetně registrů a podobně. Dále se emuluje paměť cílové platformy a ostatní hardware. I přes různé optimalizace se jedná o nejméně efektivní způsob virtualizace. Na druhou stranu je to jediný způsob jak virtualizovat jinou architekturu. Lze i emulovat mnohaprocesorový stroj na počítači s jedním procesorem apod.

K nejznámějším emulátorům patří BOCHS, PearPC, QEMU. Emulátory jsou rovněž často využívány hráči počítačových her pro již dnes historické plaftormy jako ZX Spectra, Atari, Amigy či notoricky známý DOSBox pro emulaci systému MS-DOS.

Hypervisor

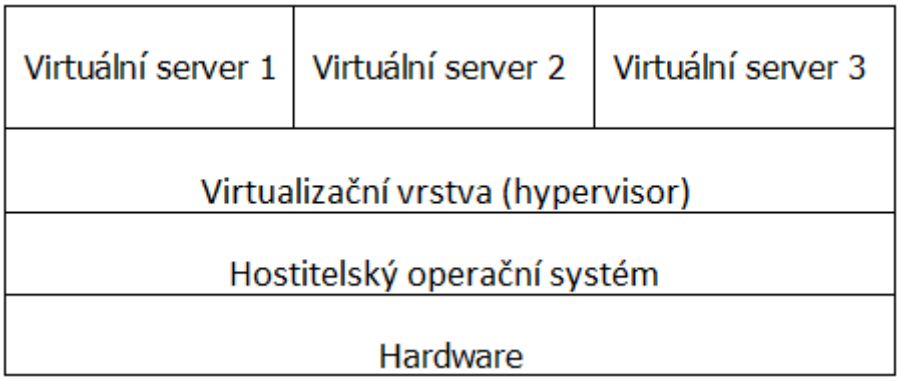
Hypervisor neboli také Virtual Machine Monitor (VMM) je označení pro jednu z technik virtualizace hardware počítače, která umožňuje na jednom počítači spustit zároveň více operačních systémů. Hypervisor je nejvyšším arbitrem, který řídí přístup virtualizovaných počítačů (tzv. hostovaný systém) k hardwaru počítače (hostitelský systém), řídí jejich běh a zároveň je od sebe odděluje. Poprvé se pojem hypervisor objevil v roce 1965 v souvislosti se softwarem IBM RPQ pro počítač IBM 360/65. To dovolilo tomuto počítači sdílet paměť – polovinu jako IBM 360, druhou polovinu jako emulovaný stroj IBM 7080. Hypervisory se nejčastěji dělí do dvou skupin, podle toho, ve které vrstvě je hypervisor umístěn:

a) *Typ 1 (nativní)* – Hypervizor typu 1 běží přímo na hostitelském hardwaru. Zde monitoruje a řídí běh virtualizovaných strojů (hostovaných systémů) z hlediska hardwarových prostředků. Hostovaný operační systém tak funguje na jiné úrovni, pod hypervizorem. Tento model představuje klasické provedení architektury virtuálního stroje, původní hypervizor byl CP/CMS, vyvinutý v IBM v roce 1960. Dnešní představitelé této koncepce jsou XEN, VMware ESX, Hyper-V. [12]



Obrázek 2.2: Hypervisor, Typ 1

b) *Typ 2 (hostovaný)* - Hypervisor typu 2 je spuštěn v konvenčním prostředí operačního systému. Vrstva hypervisora se nachází nad vrstvou operačního systému. Nejznámější představitelé používající tento typ hypervisoru jsou KVM, VMware Workstation, VirtualBox, Virtual PC & Virtual Server. [12]



Obrázek 2.3: Hypervisor, Typ 2

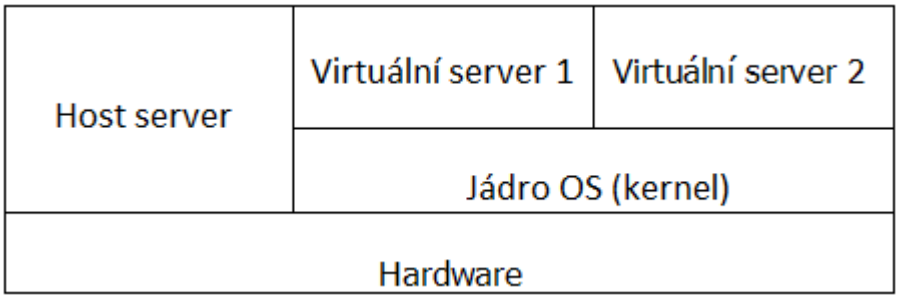
2.1.2 Softwarová virtualizace

Na úrovni softwarové virtualizace se blíže zmíním o dvou důležitých představitelých této kategorie, tj. virtualizace na úrovni jádra OS a aplikační virtualizace.

Virtualizace na úrovni jádra operačního systému

Koncepcí této metody virtualizace je využití jediného jádra operačního hostitelského systému několika izolovanými virtuálními stroji (často se v terminologii používá „kontejnery“). Je tedy zřejmé, že hostitelský i hostované systémy musí být stejné (dokonce ve stejné verzi). Jedinou výhodou hostitelského stroje je, že má možnost přidělovat fyzické zdroje hostovaným systémům. V ostatních ohledech jsou stroje rovnoprávné - k přístupu k hardware využívají týž kernel. Díky tomu je výkonnostní režie tohoto řešení prakticky nulová. Nevýhoda tohoto řešení tkví v tom, že

chyby v hostitelském systému mají fatální následky, jelikož se dále duplikují na všechny virtuální stroje (kontejnery). Typickými představiteli tohoto typu virtualizace jsou Jail ve FreeBSD, Containers v Solarisu, OpenVZ.



Obrázek 2.4: Virtualizace na úrovni jádra operačního systému

Aplikační virtualizace

Aplikační virtualizace popisuje technologii, která zlepšuje přenositelnost, škálovatelnost, ovladatelnost a kompatibilitu aplikací, tím, že je zapoudří od operačního systému, kde jsou tyto aplikace spouštěny. Takováto aplikace věří, že komunikuje přímo s operačním systémem a všemi jeho prostředky, ale ve skutečnosti tomu tak není. Tento typ virtualizace dovolí aplikaci běžet pod jiným, než jejím nativním prostředím. Výhodou aplikační virtualizace je, že využívá méně systémových prostředků než virtualizace s celým OS, na druhé straně ne všechen software může být takto virtualizován. Známými představiteli této kategorie je aplikace Wine, která dovoluje aplikacím napsaným pod MS Windows, aby byla spuštěna na unixových systémech a aplikace Cygwin, která zase naopak umožňuje spouštět unixové aplikace na systému Windows.

2.2 Hlavní představitelé virtualizace

VMware

Patrně se jedná o nejznámější firmu na virtualizačním poli. Společnost byla založena v roce 1998 a od té doby expandovala tím způsobem, že dnes nabízí několik typů virtualizačních systémů. VMware virtualizační software lze nainstalovat na OS Windows, Linux a Mac OS X, zatímco serverové virtualizační řešení používá hypervisor, který běží přímo na hardwaru. Nejznámějšími produkty této společnosti jsou VMware Workstation a VMware EXS pro serverové enterprise řešení.

VMware Workstation – VMware Workstation je nejznámější a nejhojněji využívaný produkt firmy VMware. Podporuje virtualizaci platformy x86 a x86-64, tzn. že lze vytvářet virtuální stroje, které používají procesory pouze s toutou sadou instrukcí. Každá takto vytvořená virtuální instance má svůj operační systém (Windows, Linux, BSD apod.). Hypervisor je typu 2, to znamená, že

hypervisor se nachází nad vrstvou operačního systému, ne nad hardwarovou vrstvou. VMware Workstation je dobře znám svou pokročilou funkcionalitou správy snapshotů.

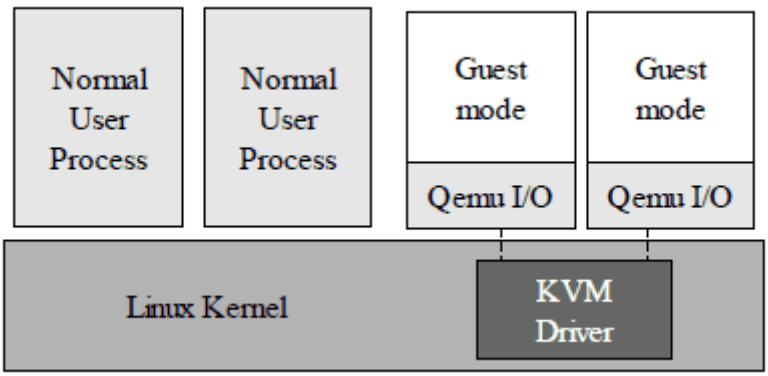
VMware EXS – VMware EXS je enteprise řešení pro podniky a firmy, které obsahuje funkcionality pro efektivní využití virtualizace ve firemní IT struktuře. Hypervisor je umístěn přímo na hardwarové vrstvě, aniž by ke svému běhu požadoval jakýkoliv operační systém. Místo toho hypervisor má svůj vlastní kernel.

Kernel-based Virtual Machine (KVM)

KVM je zástupce tzv. plné virtualizace. To znamená, že virtuálnímu počítači je k dispozici virtuální procesor, grafická karta, síťová karta a další periferie. Ty se poté spojují s reálným hardwarem pomocí určitého rozhraní.

KVM je virtualizační nástroj pro Linuxové distribuce. Původně byl podporován na architekturách x86 a x86-64, poté se rozšířila podpora i pro další architektury jako S/390, PowerPC a IA-64. KVM podporuje virtualizaci celé řady operačních systémů přes různé linuxové distribuce, BSD, Solaris, Windows, Haiku a ReactOS. Toto virtualizační řešení je přímou součástí Linuxu 2.6.20 a vyšších verzí. Je možno ho používat na procesorech, které jsou vybaveny technologií Intel VT nebo AMD-V. KVM je licencováno pod GNU General Public Lincense, tzn. že se jedná o nekomerční nástroj.

V KVM je každý virtuální stroj realizován jako regulérní linuxový proces. Klasický linuxový proces má dva režimy spouštění – uživatelský a kernel režim. KVM přidává třetí – hostovaný režim (který má vlastní kernel a vlastní uživatelské režimy). Součástí KVM je emulátor zvaný QEMU, který emuluje procesor a další nezbytné periferie počítače. [13]



Obrázek 2.5: KVM – schéma virtualizace

Virtual Box

Virtual Box je virtualizační nástroj vytvořený společností Innotek GmbH, poté zakoupený společností Sun Microsystems a v současné době vyvíjený Oracle corporation jako součást její

rodiny virtualizačních produktů. Virtual Box je instalován na hostujícím operačním systému jako bežná aplikace. Takto nainstalovaná aplikace následně dovolí vytváření virtuálních strojů s hostovanými operačními systémy, každý ve svém vlastním virtuálním postředí.

Virtual Box podporuje celou řadu 32 a 64 bitových hostujících operačních systémů jako Linux, Mac OS X, Windows, Solaris a BSD. Podporované hostované operační systémy zahrnují různé verze a distribuce operačního systému Windows, Linux, BSD, OS/2, Solaris a další. Virtual Box podporuje hardwarovou virtualizaci pro platformy Intel VT a AMD-V, což umožňuje plnou virtualizaci s podporu hardwaru.

Microsoft Virtual PC

Microsoft Virtual PC 7 (nástupce Microsoft Virtual PC 2007, Microsoft Virtual PC 2004) je virtualizační program pro Microsoft Windows. Nejnovější verze programu, Microsoft Windows Virtual PC 7, podporuje jen hostitelské operační systémy Windows 7. Hostované operační systémy mohou být edice Windows 7, Windows Vista či Windows XP Professional. Ostatní operační systémy (jako např. Linux) nejsou podporovány.

Microsoft Virtual PC 7 obsahuje funkcionalitu zvanou *Windows XP mode*, která uživatelům Windows 7 dovoluje spouštět aplikace pod Windows XP. Typický scénář je takový, že firma, která chce v napříč celou organizací začít používat Windows 7, má stále některé důležité aplikace navrženy pouze pro použití v systému Windows XP a nelze je provozovat v systému Windows 7. To však nepředstavuje překážku, pokud firma používá Windows XP mode. Ten lze používat dvěmi způsoby:

a) *jako samostatný virtuální počítač* – Virtuální prostředí systému Windows XP lze využívat se všemi výhodami, jako je podpora rozhraní USB, přístup k síti, přidružení souborů či sdílení schránky.
b) *jako integrované prostředí* – Aplikace nainstalované ve virtuálním počítači jsou v hostitelském počítači k dispozici pomocí zástupců v nabídce Start.

Hyper-V

Microsoft Hyper-V, v minulosti nazýván jako Windows Server Virtualization, je virtualizovaný serverový systém pro systémy x86-64. Hyper-V je hypervizorově stavěný serverový systém, což znamená, že má svůj vlastní hlavní operační systém (většinou Windows Server 2008) a pomocí virtualizace se skrze něj mohou spustit další operační systémy. Hypervisor leží přímo na hardwarové vrstvě, tudíž se jedná o hypervisor typu 1 – nativní. Hyper-V dnes existuje ve dvou variantách: Jako samostatný produkt, nazývaný Microsoft Hyper-V Server 2008 nebo jako instalovatelná součást operačního systému Windows Server 2008. Podporované hostované operační systémy jsou Windows Server 2008, 2003 a 2000, Windows 7, Windows Vista, Windows XP, z linuxových distribucí je to SUSE Linux a Red Hat Linux a pak ještě CentOS. Hyper-V je virtualizační řešení pro podniky a velké společnosti, které soutěží s konkurenčním řešením VMware ESX.

Hypervisor musí mít přinejmenším jednu rodičovskou instanci, tj. běžící OS Windows Server 2008. Samotná virtualizace probíhá v rodičovské instanci a má přímý přístup k hardarovým zařízením. Rodičovská instance pak vytvaří hostované operační systémy použitím tzv. hypercall API, což je aplikační rozhraní vystavené Hyper-V. Hostované operační systémy nemají přímý přístup k fyzickým zdrojům, místo toho vidí mapu virtuálních zařízení. Jakákoliv žádost takového virtuálního zařízení je přesměrována přes sběrnici VMBus k zařízení v rodičovské instanci, které je již fyzické postará se o zpracování žádosti. Odpověď je opět přesměrována přes VMBus, což je logický kanál umožňující komunikaci mezi rodičovskou instancí a hostovaným OS.

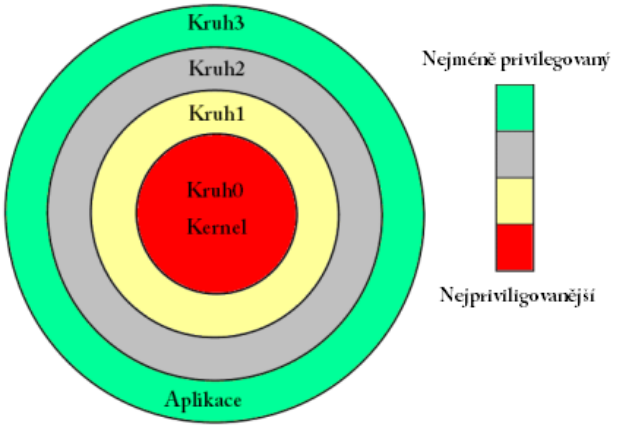
2.2.1 Xen

Virtualizační nástroj Xen popíšu podrobněji a více do hloubky, jelikož je to nástroj, který jsem si vybral pro virtualizaci PC a který bude použit ve školním systému Virtlab. Xen je hypervisor poskytující rozhraní pro virtualizaci hardware a běh více operačních systémů na jednom počítači současně.

První verze Xenu vznikla v laboratořích na univerzitě v Cambridge v roce 2003. Původní autoři Xenu jsou Keir Fraser a Ian Pratt. V roce 2007 proběhla akvizice ze strany společnosti Citrix Systems. V roce 2009 Xen zareagoval na roustoucí popularitu Cloud computingu a uvolnil projekt Xen Cloud Platform (XCP), který Xen přenaší do cloudového prostředí. Od roku 2010 je Xen komunitním projektem publikovaným pod licencí GPL jako open source. Xen se vyvíjí pro platformy IA-32, x86-64, Itanium a ARM. [4]

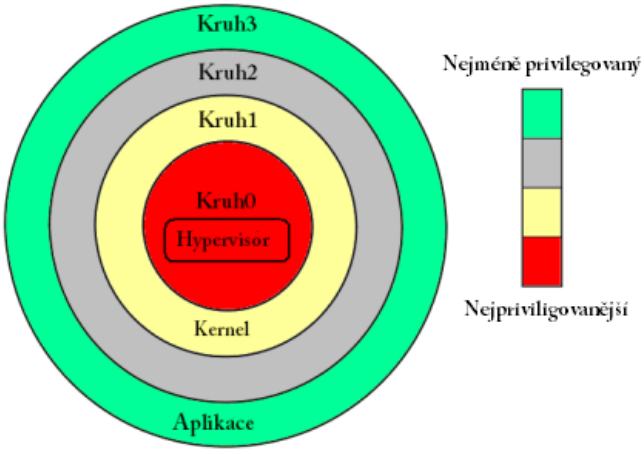
Začlenění Xenu do platformy x86

Současné procesory dokáží pracovat v různých režimech z důvodu ochrany proti neautorizovanému přístupu k fyzickému procesoru a hardware (reálný, privilegovaný, nereálný mód aj.). Z pohledu virtualizace je nejdůležitější prvilegovaný režim procesoru. Tento režim vytváří tzv. chráněné úrovně, označované jako kruh (ring). Procesory x86 rozeznávají čtyři kruhy, které jsou číslovány od 0 do 3. Nejnižší, kruh 0, je označováno jako „supervisor mode“ a je to úroveň s největšími privilégii vzhledem k hardwarovým prostředkům. Na této úrovní běží jádro OS, správa paměti, kontrola přístupu k hardwaru a privilegované instrukce, tzn. instrukce, které mohou ohrozit běh systému. Kruh 1 a 2 se používají minimálně, kruh 3 je onzačovaný jako „user mode“ a je určený pro běh neprivilegovaných instrukcí, tj. uživatelských instrukcí. [13]



Obrázek 2.6: Úrovně privilegovaného režimu procesoru x86

Při virtualizaci prostřednictvím virtualizačního nástroje Xen dochází k několika změnám oproti předcházejícímu popisu. Jádro OS neběží v supervisor módu, ale na méně privilegované úrovni v závilosti na tom, či je použitá 32bit. architektura (kruh 1) nebo 64bit. (kruh 3). Privilegované instrukce však na úrovní kruhu 1 (respektive 3) nelze spouštět, proto Xen zavádí vrstvu označovanou jako hypervisor, která je odpovědná za vykonávání privilegovaných instrukcí v kruhu 0 a je jakýmsi zprostředkovatelem mezi fyzickým hardwarem a virtuálními stroji Xenu.



Obrázek 2.7: Xen a jeho začlěnění v privilegovaných úrovních režimu

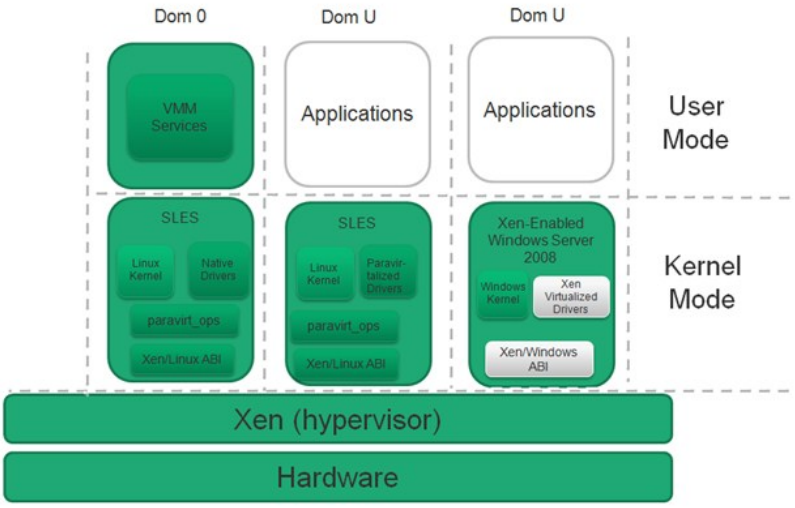
Architektura Xenu

Virtualizační prostředí Xenu lze rozdělit na tři hlavní části, které dohromady tvoří kompletní virtualizační platformu:

1) *Xen hypervisor* – Jedná se o základní abstraktní vrstvu softwaru, která leží přímo na fyzickém harwaru, tudíž je položena níže než operační systém. Tato vrstva je zodpovědná za plánování procesorového času a přerozdělování paměti virtuálním strojům. Hypervisor se nestará pouze o abstrakci hardwaru, ale také řídí běh virtuálních strojů, jak sdílejí společné pracovní

prostředí. Hypervisor nemá žádnou znalost síťového prostředí, externích disků, videa či jakýkoliv jiných běžných I/O funkcí.

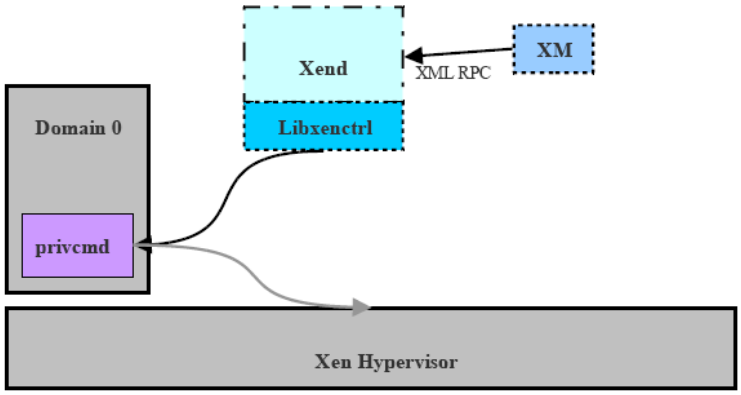
2) *Doména 0, privilegovaná hostitelská doména (Dom0)* – Doména 0 musí běžet předtím, než je spuštěn jakýkoliv virtuální stroj. Je spuštěna hypervisorem hned během úvodní start-up fáze systému. Je to hostitelský operační systém s upraveným linuxovým jádrem kernelu. Dom0 je specifický virtuální stroj, který má jako jediný speciální práva k přístupu k fyzickým I/O zdrojům, stejně jako je zodpovědný za komunikaci s ostatními virtuálními stroji. Po přihlášení do domény 0 je možno spravovat celý systém včetně všech virtuálních PC. Podporované hostitelské systémy jsou již dnes v podstatě všechny linuxové distribuce.
3) *Doména U, neprivilegovaná hostovaná doména (DomU)* – Doména U je spouštěna a řízena Doménou 0 a beží na ní speciálně modifikovaný hostovaný operační systém. DomU nemá přímý přístup k fyzickému hardwaru tak jako Dom0 a je často označovaná jako neprivilegovaná. DomU komunikuje s Dom0 prostřednictvím volání zvaných hypercall ABI. Jedná se o softwarová volání, které jsou reakcí např. na potřebu vykonání privilegovaných operací. Hostované domény můžou být dvojího typu:
    a) *Paravirtualizace* – Hostovaná doména s paravirtualizací ví, že nemá přímý přístup k fyzickému hardwaru a rozeznává ostatní virtuální stroje, které jsou běžící nad stejným hypervisorem. Všechny takto paravirtulizované virtuální stroje používají modifikovaný Linux operační systém, který si je vědom, že je spuštěn na hypervisoru místo na fyzickém hardwaru, tzn. že nespouští privilegované operace (procesor, paměť), ale v případě potřeby vykonání těchto operací volá hypervisor pomocí hypercallů. Linuxových distribucí, které mohou být použity a upraveny pro Doménu U je celá řada od SUSE, Red Hat, Fedora, Ubuntu, Gentoo, Solaris až po OpenBSD a další.
    b) *Hardware Virtual Machine (HVM)* – Hostovaná doména typu HVM je použita pro virtualizaci systémů Windows či jakýkoliv jiný nepozměněný OS. Jedná se o hardwarovou virtualizaci (Intel VT, AMD-V), kterou musí daný procesor podporovat, aby mohla být tato virtualizace použita. Takto virtualizovaný operační systém není nijak modifikován a není si vědom toho, že běží ve virtualizovaném prostředí a sdílí fyzické zdroje spolu s dalšími virtuálními stroji. V tomto případě je spuštěn emulátor QEMU, který emuluje fyzické prvky počítače. [4]



Obrázek 2.8: Architektura Xenu

Ovládací prvky Xenu

Xen používá sérii linuxových démonů, kteří se starají o celkovou správu a řízení virtualizačního prostředí a tito démoni jsou umístěni v Doméně 0. Diagram níže ukazuje, jak tito démoni mezi sebou spolupracují. Xend démon je aplikace napsaná v jazyce Python a je to systémový manager pro prostředí Xenu. Xend obsahuje knihovnu Libxenctrl, napsanou v jazyce C, která komunikuje s hypervisorem přes Doménu. Speciální ovladač uvnitř Domény 0 - privcmd pak doručuje žádosti do hypervisoru. XM je command line tool, který přijímá uživatelské vstupy a posílá je démonu Xend přes protokol XML-RPC.



Obrázek 2.9: Schéma ovládacích prvků Xenu

2.3 Výhody a nevýhody virtualizace

Virtualizace přináší spoustu nesporných výhod, díky kterým se z virtualizace stal v poslední dekádě takový fenomén, ale samozřejmě má také své stinné stránky a je nutné pečlivě zvážit v jakých situacích převažují klady virtualizace a v jakých spíše zápory. Hlavní klady a zápory virtualizace jsou podrobněji rozepsány v následujících odstavcích.

Výhody virtualizace

- *Konsolidace serverů* – Vzhledem ke komplikovanosti a komplexnosti moderních SW produktů se již delší dobu ustálil trend v jejich odděleném provozování na samostatném HW. Je běžné provozovat samostatné databázové, aplikační, poštovní servery, souborové servery, webové servery apod. To vede k potřebě velkého množství PC serverů se všemi důsledky - náročnou systémovou podporou, nižším využitím výkonu jednotlivých strojů, nároky na místo a v neposlední řadě na potřebnou elektrickou energii, klimatizaci a záložní zdroje napájení. Navíc je známý fakt, že průměrná zátěž průměrného serveru činí zhruba 15%**.** Zbylých 85% je tedy nevyužitých. Konsolidace serverů tento poměr obrací. Konsolidací se rozumí provozování většího počtu virtuálních serverů na menším počtu serverů fyzických. Tím dosáhneme značné úspory v prostorách, elektrické energie, nákladu na HW podporu vzhledem k menšímu počtu fyzických serverů a potažmo ke značným finančním úsporám. Díky těmto úsporám se virtualizace řadí do tzv. Green IT.
- *Konsolidace historických serverů* – Virtualizace je velmi užitečná v případě zastaralých aplikací a serverů, které je z historických důvodů nutno stále udržovat v chodu. Takové aplikace nemusí správně fungovat na novém hardwaru či operačním systému. Pokud ano, tak tyto aplikace zdaleka nevyužívají potenticál nového výkonného hardwaru a tak se zde opět nabízí možnost z předchozího bodu, tj. konsolidace takovýchto serverů.
- *Minimalizace vzájemného ovlivňování provozovaných systémů, zapouzřední* – Virtualizace umožňuje dále rozvinout trend specializace jednotlivých serverů, instalovaný SW má tak „svoje“ provozní prostředí a není negativně ovlivňován jiným instalovaným SW v rámci téhož systému. Další možností je spouštět v takto izolovaném systému nedůvěryhodné aplikace, které zde nemůžou způsobit větší škodu.
- *Zálohování, migrace* – Virtuální počítač je velmi snadné zálohovat, jelikož bežně obsahuje jeden konfigurační soubor a diskové soubory tvořící emulované diskové jednotky virtuálního počítače. Těchto několik souborů stačí zkopírovat a záloha systému je hotova. Rovněž migrace je díky virtualizaci velice snadná a efektivní.
- *Rychlejší a pružnější obnova po výpadku* – Virtuální stroje právě běžící na havarovaném fyzickém serveru lze velmi rychle (a u nových virtualizačních SW dokonce automaticky) během okamžiku spustit znovu na jiném fyzickém serveru.
- *Vývoj, testování a prezentace* – Pokud se vyvíjí a testuje software, u kterého je třeba ověřit, že je kompatibilní s více OS jako např. Windows 2000, Windows XP (včetně všech variant Service Packů), případně na Windows Server 2003, je virtualizace ideální řešení, kdy se pro každou požadovanou verzi příslušného operačního systému vytvoří virtuální počítač s tímto OS. Takové testovací prostředí lze navíc snadno kdykoliv znovu obnovit

„klonem“ původního prostředí. Případně pokud je třeba prezentovat produkt, který má speficické (serverové) požadavky a nemůže být na klasickém PC spuštěn, pak opět na scénu nastupuje virtualizace.

- *Rychlá implementace nových serverů* – Při provozování standardních fyzických serverů vyvolá náhrada starého provozního serveru novým velké množství práce - je třeba instalovat nejenom operační systém a nově všechny aplikace, ale kvůli tvorbě zcela nového prostředí je třeba provést velké množství testů, zda je nové prostředí stoprocentně kompatibilní a funkční. V případě virtualizace stačí na nový fyzický server instalovat jen virtualizační prostředí. Provozní virtuální stroje pak stačí na nový server jenom přenést. Lze dosáhnout úspory pracnosti a času v řádu dnů až týdnů.

Nevýhody virtualizace

- *Zvýšení rizika nedostupnosti služeb či ztráty dat* - V případě, že na fyzickém serveru běží např. pět virtuálních serverů a fyzický server zhavaruje, tak všech pět těchto virtuálních serverů se stane nefunkční. Tím hůře, pokud na fyzickém serveru „ztratíme“ data. Řešení této nepříjemné situace je několik. Můžeme provádět pravidelné zálohy systému. Můžeme také zřídit redundantní stroj, který se spustí v případě havárie primárního stroje. Tato protiopatření však eliminují výhody virtualizace.
- *Snížení výkonu* – Není pochyb, že virtualizace vyžaduje výkonný hardware, pokud má na něm běžet několik virtuálních PC. Největší zádrhel však je správně analyzovat hardwarové potřeby pro virtualizaci. Existují sice rozličné návody a nástroje pro takovou analýzu, každý sofware se však chová jinak ve virtualizvaném prostředí. Je známo, že některé aplikace, které mají skromné hardwarové požadavky, mají ve virtualizovaném prostředí několikanásobně vyšší požadavky.
- *Vyšší riziko odcizení dat* – Pokud se nezvanému hostu podaří získat přístup k fyzickému serveru, kde beží několik virtuálnch serverů, získává automaticky přístup ke všem virtuálním serverům.
- *Nedostatečná podpora aplikací* – Ne všechny aplikace plně podporují virtualizaci, např. některé klíčové aplikace jako databázové systémy. Občas se aplikace či operační systém ve virtuálním stroji může chovat zcela jinak než na fyzickém stroji bez jakékoliv zjevné příčiny. Pokud k takové chybě dojde, je mnohem obtížnější ji lokalizovat.
- *Zvýšené licenční nároky a složitost* - pro provoz virtuálních systémů platí různá licenční pravidla - některá jsou stejná jako pro fyzické systémy, některá odlišná. Celý proces licencování se v každém případě stává složitějším.
- *Správa, údžba a zaplátování virtuálních OS* – Fakt, že jde o server virtuální, neznamená, že jej není třeba zahrnout do celkové správy a údržby. Je třeba mít i tyto úkony maximálně automatizované a konfigurovatelné mimo samotné virtuální servery – vhodnými nástroji umožňujícími automatizaci této činnosti.
- *Exploze virtuálních serverů* (z angličtiny pojem virtual machine sprawl) – Ačkoliv správa virtuálních strojů je docela komplexní záležitost, vytvoření nového virtuálního stroje je velice snadné – stačí naklonovat obraz virtuálního počítače do nového virtuálního stroje a

je hotovo během několik sekund. Tento fakt obvykle vede k velkému nárustu virtuáních strojů, které je pak velmi obtížné spravovat. Některé z těchto strojů byly vytvořeny pouze z důvodu, že je tak snadné a rychlé vytvořit nový virtuální stroj.

- *Potřeba přenesení stávajících fyzických prostředí do virtuálních strojů* – Tento jednorázový proces lze podpořit použitím specializovaného (tzv. P2V – physical to virtual) software.

3 Analýza požadavků a návrh systému

3.1 Zadání, specifikace

Cílem této diplomové práce je rozšířit funkcionalitu školního systému Virtlab o možnost uchovávání obsahu souborového systému uživatelských PC realizovaných v tomto systému a jeho převodu mezi jednotlivými rezervovanými úlohami, tzn. poskytnout uživatelům možnost přerušované práce na témže problému při zachování konfigurace.

Virtlab je distribuovaný systém, který na základě studentova požadavku ověří dostupnost žádaných síťových prvků, které se nutně nemusí nacházet na jednom místě (vzdálené lokality), a za předpokladu, že jsou tyto síťové prvky dostupné, vytvoří z nich na určitý časový interval počítačovou síť, se kterou student může pracovat. Takto vytvořená počítačová síť by také určitě neměla postrádat koncové uživatelské stanice, tak aby si student mohl ověřit některé charakteristiky a chování dané sítě. Nabízí se přímé zapojení fyzických koncových počítačových stanic do takto vytvořené počítačové sítě. Tato možnost však už na první pohled není efektivní a už vůbec ne snadno implementovatelná. V současné době, která skýtá přehršel pokročilých virtualizačních nástrojů pro virtualizaci počítačových stanic, se řešení samo nabízí. Proto vznikla tato diplomová práce, která systém Virtlab o zmíněnou funkcionalitu virtuálních koncových uživatelských stanic rozšiřuje.

Dříve, než se započne se samotnou implementací je nejprve nutné provést několik kroků. První z nich je určitě prozkoumat a blíže se seznámit se systémem Virtlab, a to jak z uživatelského hlediska, tak i z hlediska architektury, tzn. jakým způsobem je v systému Virtlab vytvořena požadovaná počítačová síť. Systém Virtlab bude podrobněji popsán v následující kapitole. Poté vybereme několik kandidátů pro virtualizaci, které se nejvíce hodí ke stávajícímu řešení Virtlabu a na základě analýzy těchto vybraných virtualizačních nástrojů společně s požadavky systému Virtlab zvolíme vítězný nástroj, který bude pro virtualizaci koncových uživatelských PC použit. Na základě vybraného virtualizačního řešení navrhneme strukturu archivace záloh virtuálních PC tak, aby si je uživatelé Virtlabu mohli uchovat i po skončení platnosti sítě a později je použít například u jiné úlohy aniž by musely tyto virtuální PC znovu vytvářet a zdlouhavě je konfigurovat. Struktura archivace záloh bude do velké míry ovlivněna virtualizačním řešením, které jsme vybrali. Každý virtualizační nástroj totiž řeší otázku snapshotů (obrazu virtuálních PC) jiným způsobem. Jakmile bude hotova implementace uschování záloh virtuálních strojů, tak celé toto řešení zakomponujeme do Virtlabu, včetně jeho grafické části, ze které se bude správa takto vytvořených snaphotů ovládat.

3.2 Virtuální laboratoř počítačových sítí (Virtlab)

Myšlenka virtuální laboratoře se vyvinula z potřeby vzdáleně zpřístupnit specializovaná zařízení, která se nacházejí ve školních laboratořích, pro výuku předmětů zabývajících se počítačovými sítěmi, zejména pro potřeby studentů, kteří nemají možnost osobně navštěvovat zmíněné laboratoře, a tak zajistit možnost dálkové práce s těmito zařízeními z Internetu. Její vznik inicioval Petr Grygárek a postupně je realizována s pomocí diplomantů, zejména inženýrského studia, na katedře informatiky. Základní koncepce systému byla definována v roce 2005.

Jak již bylo zmíněno, smyslem Virtlabu je zpřístupnit laboratorní prvky pro praktickou výuku počítačových sítí vzdáleně prostřednictvím Internetu. Studenti si mohou pomocí www rozhraní rezervovat laboratorní prvky na určitý časový interval a následně k nim přistupovat pomocí běžného internetového prohlížeče s podporou Java appletů. Propojení laboratorních prvků se uskuteční automaticky podle výběru konkrétní úlohy ze souboru nabízených laboratorních úloh, nebo si student může zadat svou vlastní topologii. Systém dovoluje spolupráci více vzdálených lokalit (laboratoří) vzájemně sdílejících síťové prvky a realizaci virtuálních síťových topologií přes Internet. Fyzické síťové prvky nutné pro vytvoření studentem vybrané síťové topologie jsou v době rezervace vyhledávány dynamicky ve všech lokalitách.

3.2.1 Architektura Virtlabu

Omezením původní verze Virtlabu byla jeho pevná fyzická topologie. Jestliže bylo potřeba změnit fyzickou topologii, musel zodpovědný člověk jednotlivé síťové prvky ručně propojit do nově požadované topologie. Tento výrazně omezující faktor se podařilo překonat pomocí zařízení zvaného ASSSK (Automatizovaný Systém Správy Síťových Konfigurací). Toto zařízení dokáže fyzicky propojovat sériové linky na základě příkazů, které jsou mu předávány. Zapojení konkrétní fyzické topologie podle potřeb studenta, lze tedy jednoduše automatizovat. Propojení ethernetových linek je realizováno technikou Q-in-Q. Tato technika je založená na technologii 802.1q, která umožňuje použití VLAN. Q-in-Q vkládá do hlavičky ethernetového rámce dvě specifická označení – první označení provádějí samotná propojovaná zařízení, druhé označení pak provede zařízení, které propojuje jednotlivé prvky (vytvoří tunel simulující trunk linku), aby odlišilo jednotlivé toky dat, které odpovídají logickým spojům mezi propojovanými zařízeními.

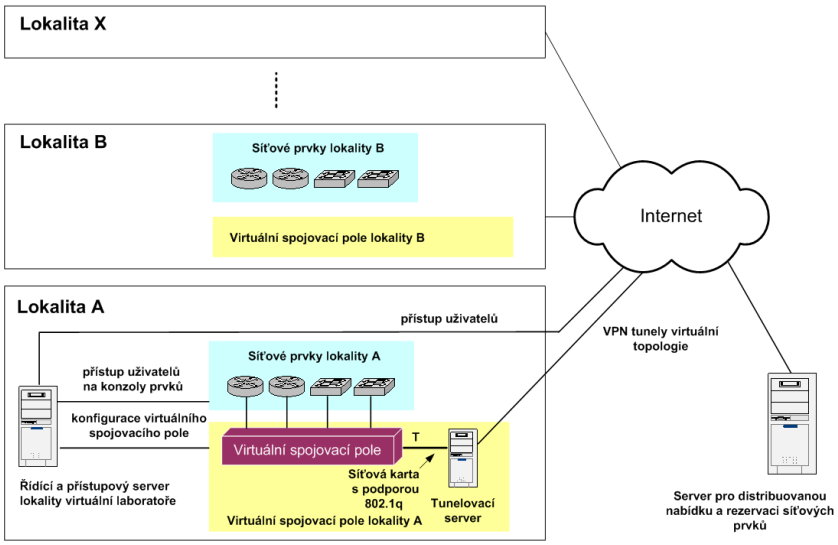
Virtlab je koncipován jako distribuované řešení, tzn. že by měl být schopen zpřístupňovat síťová zařízení síťových laboratoří v různých lokalitách (napříč světem). Systém rozdělen na základní jednotky, které jsou nazvány lokality, a které reprezentují jednotlivé síťové laboratoře. Každá lokalita má svá pravidla, jež definují, která zařízení a ve kterém čase jsou k dispozici vybraným vzdáleným lokalitám. V každé lokalitě proto běží Rezervační server, který obhospodařuje práva uživatelů Virtlabu a zabezpečuje zapůjčování sítových prvků prvků.

Jestliže uživatel požaduje vytvořit jistou úlohu, musí vytvořit návrh žádané síťové topologie. Ten neobsahuje přímo fyzická zařízení, ale určuje pouze typy síťových zařízení a způsob jejich propojení, proto jej nazýváme logická topologie. Teprve až je vytvořena tato logická topologie, zjišťuje se, je-li v daném čase možno realizovat také její fyzickou formu. Poté se

vyberou vhodná síťová zařízení - a to jak lokální, tak i vzdálená, a prostřednictvím Rezervačních serverů se tyto prvky pro danou topologii vyhradí.

Při započetí definované rezervace vyvolá Rezervační server prostřednictvím aktivačního skriptu generování konfigurací topologie, příslušné k dané rezervaci, pro virtuální spojovací pole a tunelovací servery (vysvětleno v následujícím odstavci). Aktivační skript poté odešle soubory s vygenerovanou konfigurací všem Konfiguračním serverům v definovaných lokalitách, jejichž úkolem je příjem konfiguračních souborů a následně také nahrání těchto konfigurací do příslušných sítových prvků (virtuální spojovací pole, tunelovací server). Tímto dochází k aktivaci žádané síťové topologie na fyzické úrovni.

Zbývá ještě vysvětlit dva pojmy – virtuální spojovací pole a tunelovací server. Ke spojení zařízení v rámci jedné lokality jsou určena automatická spojovací zařízení - virtuální spojovací pole, která dokáží samočinně spojit libovolná sériová rozhraní i ethernetové porty. V současné době je pro tyto účely používáno zařízení ASSSK. Tunelovací server je pak zodpovědný za vzdálené propojení dvou segmentů distribuovaného virtuálního spojovacího pole, který ze dvou těchto různých segmentů v různých sítích VLAN udělá jeden logický ethernetový segment. Takto můžeme propojit například jeden ethernetový port na přepínači v jedné lokalitě s ethernetovým portem směrovače v jiné lokalitě, přičemž tato zařízení pracují, jakoby byla zapojena v jediném zdánlivém ethernetovém segmentu, přestože je každé zařízení v jiné lokalitě a dokonce v jiné VLAN. [3]



Obrázek 3.1: Architektura systému Virtlab

3.3 Virtualizační nástroj pro Virtlab

Tato kapitola nás provede výběrovým řízením na vhodný virtualizační nástroj pro systém Virtlab. O výběru určitého virtualizačního řešení dnes rozhoduje spousta faktorů např. zda je virtualizace dostatečně výkonná, spolehlivá, přenositelná, škálovatelná, jaké podporuje hostované

a hostitelské operační systémy, na jakých platformách lze virtualizovat, jakou má podporu a uživatelskou komunitu, jak řeší otázku snapshotů, zda je placená vs. open source a spoustu dalších měřitelných i neměřitelných parametrů.

Jak lze vidět z odstavce výše, kritérií pro výběr virtualizačního řešení je celá řada a je velmi obtížné se zaměřit na všechny. Proto si kritéria zprioritizujeme a vybereme pouze ty, které jsou z našeho hlediska nejpodstatnější. Nebudu však již popisovat žádný virtualizační nástroj, to bylo úkolem minulé kapitoly, pouze zmíním vlastnosti daného virtualizačního nástroje z hlediska našich kritérií.

Jedním z nejpodstatnějších kritérií je určitě to, jaké platformy virtualizační nástroj podporuje. Pro nás je podstatná platforma x86, kterou podporují všichni kandidáti zmínění v minulé kapitole. Na platformu přímo navazuje další důležitý faktor, tj. podpora operačního systému a to jak hostitelského, tak i hostovaného. Na hostovaném systému prakticky nezáleží, může to být jakákoliv linuxová distribuce či Windows. U hostovaného systému je podstatné pouze, aby byl počítač připojitelný do studentem vytvořené počítačové sítě přes ethernetové rozhraní, což nepředstavuje žádnou překážku, jelikož emulaci síťových karet dnes bez problémů zvládá jakýkoliv virtualizační nástroj. Odlišná situace nastává u hostitelského systému. Systém Virtlab je odladěn na systému Linux, kde také běží. Pokud virtualizační server poběží na stejném serveru jako Virtlab, pak je nezbytné, aby virtualizační nástroj podporoval hostitelský operační systém s linuxovou distribucí. Pokud virtualizační server poběží na odděleném fyzickém serveru, pak toto kritérium nemá žádný význam. V současné době je pro virtualizační server vyhrazen samostatný fyzický server, ale pokud by se situace kdykoliv v budoucnu změnila, je určitě lepší být mezi systémem Virtlab a virtualizačním serverem „kompatibilní“, to znamená, aby hostitelský operační systém mohl být některou z linuxových distribucí. Z tohoto důvodu můžeme vyřadit kandidáty Microsoft Virtual PC či Hyper-V, což jsou virtualizační nástroje od společnosti Microsoft, které podporují pouze Windows jako hostitelský operační systém.

Jelikož tato diplomová práce nepočítá s využitím komerčních nástrojů bude potřeba se zaměřit na virtualizační nástroje, které nejsou placené, tzv. open source. VMware je na poli virtualizace nejznámější komerční nástroj, takže toto řešení ihned zamítneme. Virtual Box sice vlastní Oracle corporation, ale jejich nástroj Virtual Box je vydáván zdarma pod licencí GNU General Public Lincense, přesněji řečeno jádro virtualizačního nástroje je zdarma. Tato verze Virtual Boxu je plně funkční, některé pokročilé komponenty však v této verzi chybí, jako například podpora pro zařízení USB 2.0. Virtual Box by bylo možné použít jako virtualizační nástroj pro Virtlab, ale vzhledem k tomu, že Oracle může tento svůj produkt kdykoliv plně zkomercializovat, tak od tohoto řešení raději upustíme.

Ve výběrovém řízení nám zůstaly dva virtualizační nástroje – Xen a KVM. Oba tyto virtualizační nástroje jsou v linuxovém světě dobře známé a oba nástroje jsou vedeny pod licencí GNU General Public Lincense. Rovněž oba tyto nástroje podporují jako hostitelský operační systém řadu linuxových distribucí. Xen patří mezi nejstarší virtualizační nástroje, vznikal v době, když ještě neexistovala hardwarová podpora virtualizace na architektuře x86, virtuální stroj tedy běží paravirtualizovaně – to znamená, že jádro hosta musí být patřičně upraveno, aby se nepokoušelo používat instrukce, které může použít pouze kód běžící v privilegovaném režimu. V současné době není začleněn v jádře linuxu s odůvodněním, že je příliš invazivní, znečišťuje

kód architektury x86 a případnými regresemi by trpěli všichni uživatelé Linuxu. Naproti tomu KVM si naopak cestu do linuxového jádra našlo poměrně rychle, první verze byla začleněna krátce po začátku vývoje do jádra 2.6.20. KVM využívá hardwarovou podporu virtualizace v procesoru hostitelského stroje, jedná se tedy o plnou virtualizaci, která nevyžaduje žádné úpravy hostovaného operačního systému. To byly ve stručnosti hlavní rozdíly mezi KVM a Xen, které nám však příliš nenapoví, který virtualizační nástroj je „lepší“, či který je vhodnější pro naše řešení. Ani jeden z těchto virtualizačních nástrojů nenabízí nativní podporu snapshotů, která je nutná pro archivaci záloh virtuálních PC, takže otázka snapshotů bude muset být řešena jiným způsobem, což bude probráno v další kapitole. Co se týče výkonnosti těchto virtualizačních nástrojů, tak je v podstatě srovnatelná, nejsou zde žádné podstatné výkonové rozdíly v práci s CPU, pamětí či disky. Tyto závěry potvrzuje i test porovnání těchto dvou virtualizačních platforem [14].

Z předešlého srovnání platforem Xen a KVM nevyplívají v podstatě žádné zásadní agrumenty proč bychom se měli rozhodnout pro tu či onu variantu, obě splňují kritéria pro virtualizaci v systému Virtlab a obě řešení teda můžeme použít. Po pečlivém zvážení jsem se nakonec rozhodnul pro variantu Xen, jelikož je to stabilní nástroj, který má na poli virtualizace dlouhou tradici a navíc se těší široké vývojové a uživatelské komunitě.

3.4 Archivace virtuálních instancí ve Virtlabu

Po analýze virtualizačního řešení vhodného pro systém Virtlab, kdy byl vybrán nástroj Xen, následuje další krok, který má navrhnout archivaci virtuálních strojů vytvořených v systému Xen, tak aby se uživatelé Virtlabu mohli později vrátit ke svým uloženým úlohám a mohli si znovu obnovit a načíst uložený obraz virtuálních PC s jejich vlastní síťovou konfigurací.

Jak již bylo zmíněno v minulé kapitole, Xen nenabízí žádné vlastní řešení pro práci se snapshoty, Xen nabízí pouze možnost uložit si kompletní obraz aktuálního virtuálního stroje, což je však nereálné pro použití v systému Virtlab, jelikož jeden takový obraz virtuálního stroje se souborovým systémem obsahuje řádově několik stovek megabytů dat. Pokud by měl každý student uloženo byť jen několik takových kompletních obrazů, tak by tohleto řešení vyžadovalo enormní diskovou kapacitu. Virtlab však má omezenou diskovou kapacitu a proto jsme nuceni najít řešení, které bude zaznamenávat pouze změny provedené ve virtuálním stroji, tj. změny jeho souborového systému.

Nabízí se nám dvě možnosti, jak dosáhnout požadované funkcinality, implementovat vlastní řešení pro práci se snapshoty, či najít nějaké již existující řešení a „doupravit“ ho podle vlastních potřeb. První řešení defacto znamená, že bychom si museli vytvořit vlastní souborový systém, který bude zaznamenávat a ukládat změny v souborovém systému virtuálního stroje a v případě požadavku pak virtuální stroj spustit i s těmito uloženými změnami. Tohleto řešení by bylo velice časově náročné, převyšující rámec této diplomové práce, proto jsem od tohoto řešení záhy upustil a zaměřil se na druhou variantu.

Po intenzivním hledání na Internetu, jsem našel dvě existující řešení, které jsou společně s XENem využívány pro práci se snapshoty. První je nástroj zvaný rsnapshot a druhým kandidátem je LVM (Logical Volume Management), což je v podstatě dynamická správa diskových oddílů.

Rsnapshot je open source nástroj pro zálohování souborových systémů a to jak lokálních, tak i vzdálených. Je napsaný v jazyce Perl a je kompatibilní s mnoha linuxovými distribucemi. Pro zálohu vzdálených serverů pracuje s protokolem SSH. Minimální velikost diskového prostoru, který rsnapshot pro vytvoření zálohy vyžaduje je velikost jedné plné zálohy plus místo pro inkrementální změny, to znamená, že při první záloze provede rsnapshot plnou kopii zálohovaného systému, při dalších zálohách pak už jen soubory, které byly od poslední zálohy změněny. Toto však není vlastnost, která by nám ušetřila diskový prostor, jelikož když si například pět studentů uloží pět rozpracovaných dosud neuložených virtuálních PC, tak se nám uloží pět samostatných plných záloh, což vyžaduje spoustu diskového prostoru.

Rsnapshot pro naše řešení tudíž nelze použít. Je třeba najít nástroj, který dokáže z jedné základní výchozí virtuální instance vytvořit nové virtuální instance pouze na základě inkrementálních změn. To znamená, že ve Virtlabu bude umístěna jedna univerzální výchozí instance a pokud student zažádá o vytvoření nového virtuálního PC, budou se ukládat pouze provedené změny oproti výchozí virtuální instanci. Tímto způsobem ušetříme obrovské množství diskového prostoru. LVM je nástroj, který tuto funkcionalitu obsahuje a je zmíněn v následující kapitole.

3.4.1 LVM (Logical Volume Management)

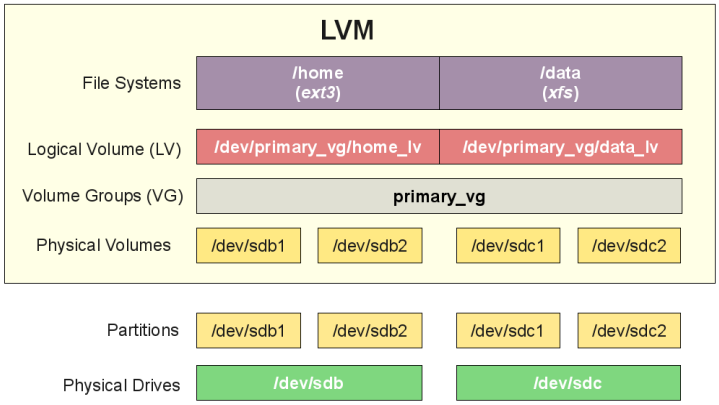
Logical Volume Management (v překladu řízení logických disků) je metoda správy diskového prostoru, která poskytuje mnohem větší variabilitu, než konvenční způsob dělení pevných disků na diskové oddíly. Umožňuje spojovat více pevných disků nebo diskových oddílů do větších logických celků (svazků) a s nimi dále pracovat, tj. vytvářet na nich oddíly se souborovými systémy (podobně jako diskové oddíly na pevných discích) a využívat jejich organizace k získání dalších vlastností (přesuny dat mezi fyzickými zařízeními bez narušení jejich dostupnosti nebo podobné vlastnosti jako mají RAID pole). Logické části je možné zvětšovat či zmenšovat v závislosti na požadavcích na zvětšování nebo zmenšování souborových systémů. Je obvyklé, že tyto operace lze provádět bez přerušení přístupu k datům v souborovém systému, pokud to samozřejmě podporuje samotný souborový systém. LVM je jeden z mnoha způsobů virtualizace diskového prostoru, která funguje jako mezivrstva mezi ovladačem pevného disku a operačním systémem.

Velká míra abstrakce, hlavní výhoda LVM, která umožňuje vytvářet libovolně velké logické disky napříč disky fyzickými, je zároveň i nevýhoda – pokud máme jeden logický disk vytvořený napříč několika fyzickými disky, tak v případě poškození jednoho z fyzických disků dojde ke ze ztrátě dat na logickém disku, pokud data nejsou zálohována či není logický disk zrcadlen. LVM má také z principu vyšší míru vnitřní (skryté) fragmentace, neboť logické disky nemusejí být alokovány v souvislých blocích, což značně zpomaluje vstupní a výstupní operace,

zvláště když je LVM postaveno na fyzických zařízení, které mají pomalé přejíždění mezi bloky dat (všechny typy konvenčních pevných disků). [15]

Princip LVM

Typicky je u LVM výchozí jednotkou fyzické zařízení (dále PV, physical volume), kterým může být celý disk nebo diskový oddíl, RAID nebo jiné zařízení pro ukládání dat. Fyzické oddíly (PV) mohou být sdružovány do svazků (dále VG, volume group). Svazky (VG) jsou směrem k operačnímu systému prezentovány jako běžná bloková zařízení. Na VG se vytvářejí logické svazky (dále LV, logical volume), které odpovídají diskovým oddílům. Na LV je pak možné vytvářet běžné souborové systémy, se kterými lze bez omezení pracovat.



Obrázek 3.2: Schéma LVM

LVM snapshot - řešení virtuálních stanic ve Virtlabu

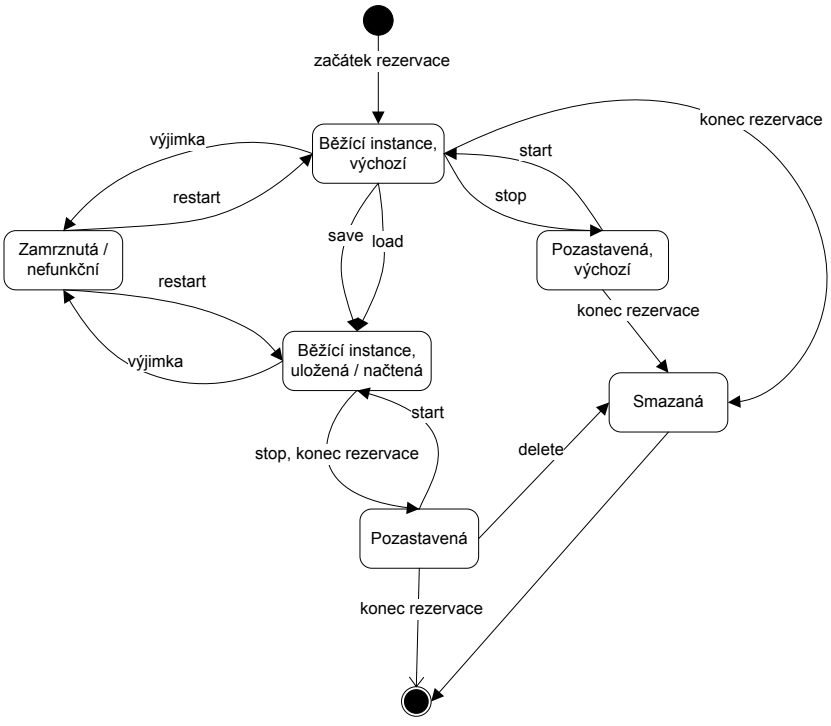
LVM dokáže vytvářet tzv. snímky (snapshot), které umožňují zachytit stav souborového systému dat v čase. Využívají se buď pro vrácení nežádoucích změn nebo pro usnadnění zálohování. V obou případech jsou nově zapsaná data ukládána do LVM takovým způsobem, aby bylo možné rozlišit původní a pozdější stav. Snímky lze později rušit či případně slučovat.

Toto je klíčová funkcionalita LVM, kterou využijeme při archivaci souborových systémů virtuálních stanic vytvořených ve Virtlabu. V LVM vytvoříme jeden logický oddíl se souborovým systémem, na který nainstalujeme speciálně upravenou linuxovou distribuci, tak aby umožňovala běh hostovaného operačního systému v prostředí Xenu. Tento hostovaný operační systém bude ve Virtlabu sloužit jako výchozí systém v případě, že si student zažádá o vytvoření nové virtuální instance. Pokud si student tuto virtuální instanci bude chtít uchovat i pro příští úlohy ve Virtlabu, tak si tuto virtuální instanci uloží. Vytvoří se snapshot výchozího hostovaného operačního systému se změnami, které student provedl během práce s virtuální instancí. Tím, že se uloží pouze provedené změny a ne celý systém, minimalizujeme nároky na diskový prostor, což je náš záměr.

3.5 Správa virtuálních instancí ve Virtlabu

V minulé kapitole bylo zanalyzováno, jak bude probíhat vytváření a ukládání virtuálních stanic ve Virtlabu v případě, že o to student zažádá. Tato kapitola naopak osvětluje, jak bude koncipována správa virtuálních stanic ve Virtlabu.

Jakmile se studentovi ve Virtlabu aktivuje jeho požadovaná virtuální síťová topologie (začátek platnosti rezervace), automaticky se také nastartují všechny virtuální koncové PC, které byly v této topologii definovány. Virtuální instance se nyní nachází v běžícím stavu a student na těchto instancích může začít pracovat. Tyto virtuální instance se však nacházejí ve výchozím stavu – mají načtený výchozí obraz systému. Pokud má student z dřívějška uložený obraz virtuální instance s vlastní konfigurací a výchozí stav systému mu nevyhovuje, je oprávněn si takovou virtuální instanci načíst. Po provedené konfiguraci nebo i během konfigurace si student může aktuální stav virtuální instance kdykoliv uložit, aby náhodou o pracně získaná data nepřišel. Pokud se během konfigurace virtuální instance vyskytne nesnáz a virtuální stanice se dostane do mrtvého bodu („zatuhne“), pak má student možnost restartu této virtuální stanice, aby se stanice dostala zpět do funkčního stavu. Po ukončení práce s virtuální instancí, může student virtuální instanci pozastavit. Za předpokladu, že by se k ní později chtěl vrátit, může takto pozastavenou virtuální instanci opět nastartovat. V případě, že student již déle pro svou práci virtuální instanci nepotřebuje, může ji odstranit. Virtuální instance však musí být pozastavená, nesmí se nacházet v běžícím stavu. Jakmile skončí platnost dané síťové topologie (konec platnosti rezervace), všechny běžící virtuální instance se automaticky zastaví. Veškerá neuložená práce na virtuálních instancích je tím ztracena. Celý tento proces přehledně ukazuje následný stavový diagram.



Obrázek 3.3: Stavový diagram života virtuální instance

I přesto, že uložené snapshoty jsou pouze rozdílem výchozí virtuální instance a studentem provedených změn a zabírají zlomkouvou velikost oproti obrazu celého systému, je třeba uživatelům Virtlabu nastavit limit, kolik místa mohou zabírat jejich virtuální instance. Výše nastavení limitu pro ukládání snapshotů je na rozhodnutí správce Virtlabu. Pokud by student uložením další virtuální instance povolený limit přesáhnul, systém to zjistí a další snapshot již nedovolí uložit. Pro tyto případy si student na stránkách Virtlabu může ověřit, kolik místa má pro snapshoty vyhrazeno a kolik volného místa mu ještě zbývá.

3.5.1 Aplikační nástroj pro správu virtuálních instancí

Po detailní analýze stavů, do kterých se během svého životního cyklu může virtuální instance dostat, přichází na řadu výběr aplikačního nástroje, ve kterém bude tato správa virtuálních stani implementována.

Na výběr máme z široké nabídky kompilačních i skriptovacích jazyků. Z velmi široké nabídky jmenuji pouze několik hlavních zástupců - C, C++, C# a Java za kompilační jazyky a PHP, Perl, Python, Ruby za skriptovací jazyky. Není pochyb o tom, že naše řešení bychom naimplementovali v každém výše zmíněném jazyce, takže analýzu a rozbor jednotlivých jazyků určitě dělat nehodlám, navíc by zřejmě měla minimální význam.

Jeden parametr zde však hraje nepoměrně větší roli a tou je přenositelnost. Z kapitoly o Virtlabu víme, že Virtlab běží pod operačním systémem Linux. Z toho plyne, že je nutné vybrat aplikační nástroj, který poskytuje podporu pro takovou platformu. To už dnes zřejmě podporuje každý aplikační nástroj, dokonce i C#, ale náš požadavek je najít takový nástroj, který nejenže bude podporovat platformu Linux, ale bude i snadno přenositelný mezi různými linuxovými distribucemi v případě, že systém Virtlab migruje na jinou linuxovou distribuci či verzi. Jedna aplikace, kterou jsem záměrně nezmínil, tento požadavek znamenitě splňuje, je to skriptovací jazyk Bash neboli také unixový shell. Tento skriptovací jazyk jsem se rozhodnul použít jako implementační nástroj pro správu virtuálních stanic v systému Virtlab nad virtualizačním nástrojem Xen se správou snapshotů v LVM. V následujícím odstavci se krátce zmíním o tomto jazku.

Bash

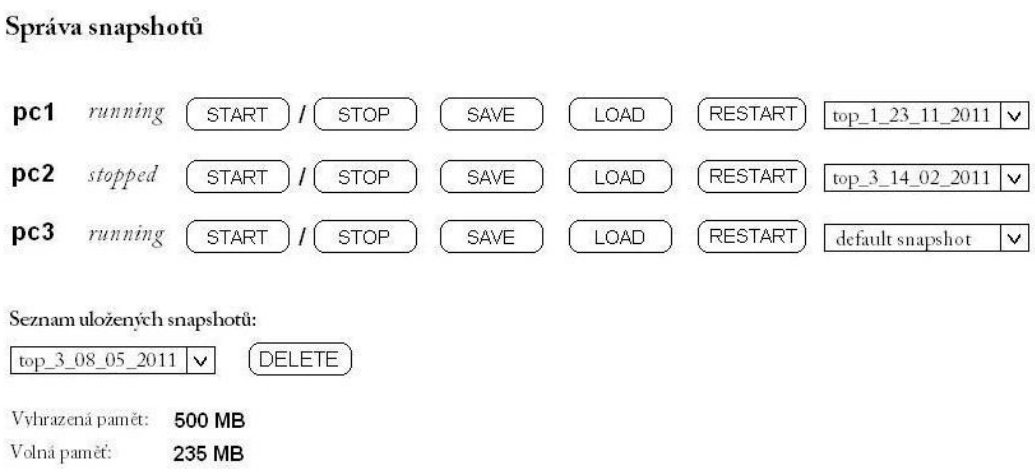
Bash je unixový shell napsaný Brianem Foxem na konci osmdesátých let. Název je akronym pro **B**ourne **a**gain **sh**ell, což poukazuje na jeho základ v dříve nejpoužívanějším unixovém shellu Bourne shell. Bash je POSIX shell s řadou rozšíření. Je koncipován pro operační systémy založené na projektu GNU a je možné ho spustit na většině unixových operačních systémů. Je používán jako implicitní příkazový interpret v systémech postavených na linuxovém jádře, stejně jako i v Mac OS X. Je možné ho použít i v systému Microsoft Windows za použití emulace POSIX pomocí softwaru Cygwin.

Základ syntaxe příkazů v jazyku Bash je převzat z Bourne shellu. Také převážná většina Bourne shell skriptů je v Bashi spustitelná bez jakékoliv modifikace kódu až na pár výjimek. Na rozdíl od Bourne shellu má Bash mnoho rozšíření i v syntaxi příkazů. Například dokáže provádět celočíselné operace bez volání externích procesů. Bash také zjednodušil přesměrování vstupů a výstupů – přesměrování standardního výstupu a standardního chybového výstupu v tu samou chvíli lze provést operátorem &>. Bash verze 3.0 podporuje regulární výrazy se syntaxí podobnou Perlu. Aktuální verze 4.0 pak podporuje asociativní pole, která umožňují vytvářet náhradu za vícerozměrná pole. [16]

3.6 Integrace do uživatelského rozhraní Virtlabu

Nacházíme se v bodě, kdy máme vybrán virtualizační nástroj, který dokáže simulovat koncové uživatelské stanice v systému Virtlab, a ve spojení se správou logických disků (LVM) si tyto koncové virtuální stanice můžeme i zálohovat v podobě snapshotů. Toto vše deleguje správa virtualizačních instancí, která je napsaná ve skriptovacím jazyku bash a která odpovídá, jak za archivaci virtuálních instancí, tak i za samotný management těchto instancí. Zbývá pouze poslední věc – zakomponovat toto řešení do uživatelského rozhraní Virtlabu.

Virtlab je webová aplikace napsaná v jazyce PHP. Pro vizualizaci jsou zde použity CSS styly a JavaScript. Před samotnou integrací s Virtlabem, si nakreslíme grafický koncept, který nám ujasní, jak by měla vypadat správa vituálních instancí ve Virtlabu, tak aby byla user friendly, tzn. uživatelsky přehledná a snadno pro uživatele použitelná a dohádatelná.



Obrázek 3.4: Grafický koncept správy virtuálních stanic

4 Implementace řešení

V minulé kapitole jsme se blíže seznámili se školním systémem Virtlab a jeho architekturou, abychom následně mohli úspěšně zakomponovat funkcionalitu virtualizovaných koncových stanic do stávajícího řešení Virtlabu. Poté jsme vybrali virtualizační nástroj Xen jako vítězného kandidáta pro virtualizaci. Pro zálohování souborových systémů těchto virtuálních stanic se ukázalo jako nejlepší řešení správa logických disků LVM s možností ukládání snapshotů. Nakonec jsme si rozepsali, jak by měla probíhat správa virtuálních stanic realizovaných v systému Virtlab a v jakém jazyce tato správa bude naimplementována. Na základě provedené analýzy, jsme získali hlavní potřebné požadavky a informace, abychom mohli začít implementovat konečné řešení virtualizovaných stanic.

Tato kapitola nás detailně provede implementací finálního řešení a poskytuje tak jasný návod, jak v systému Virtlab realizovat správu a chod virtuálních stanic krok za krokem. Jelikož Virtlab je implementován na operačním systému Linux, bylo dohodnuto, že řešení virtualizovaných stanic bude rovněž implementováno v systému Linux. Poté, co jsme úspěšně nainstalovali systém Linux, je nutné nainstalovat všechny potřebné balíčky pro aplikace Xen a LVM. Dalším následujícím krokem je vhodně nakonfigurovat aplikaci Xen pro naše potřeby a to i včetně síťového nastavení. Ve chvíli, kdy máme nakonfigurovaný hostitelský operační systém, si vytvoříme speciálně modifikovaný hostovaný operační systém, který je nezbytný pro korektní běh virtuální instance Xenu. Jakmile v Xenu úspěšně spustíme virtuální instanci, nic nám nebrání si v praxi ověřit, že je skutečně možné pomocí nástroje LVM vytvářet snapshoty virtualizovaných stanic. Vybaveni těmito znalostmi, nyní můžeme začít implementovat řešení samotné správy virtualizovaných stanic ve Virtlabu, které bude implementováno ve skriptovacím jazyce Bash. Takto naimplementované řešení propojíme se systémem Virtlab. Virtlab je aplikace napsaná v jazyce PHP, která se ovládá v prostředí webu, tudíž správa a chod virtuálních stanic budou rovněž studenty obsluhovány z tohoto místa. Nakonec zůstává nezodpovězena pouze jedna poslední otázka – jak zpřístupnit konzole virtualizovaných koncových stanic, tak aby studenti mohli pracovat s virtálními instancemi bez omezení.

4.1 Instalace potřebných komponent

Tato kapitola shrnuje seznam potřebných nástrojů a aplikací pro úspěšnou implementaci řešení správy virtualizovaných stanic.

4.1.1 Instalace Linuxu

Systém Virtlab v současné době běží nad distribucí Linuxu zvanou Debian, verze 6 (squeeze). Řešení správy virtualizovaných stanic v systému Virtlab bylo vyvíjeno pod verzí

Debian 5 (lenny). Řešení je kompatibilní s verzí Debianu 5 i 6. Ve finálním řešení běží správa virtualizovaných stanic na samostatném serveru, který je separován od Virtlabu. Je zde nainstalován operační systém Ubuntu 11. Úlohy hostitelského operačního systému se tedy nakonec zhostil Ubuntu. I zde nenastaly žádné komplikace s kompatibilitou přenesením funkcionality. Samotná instalace Linuxu probíhá klasicky podle instalačního průvodce bez jakýchkoliv speciálních zásahů do instalace.

4.1.2 Instalace Xenu

V době vývoje správy virtuálních stanic jsem používal verzi Xenu 3.2.1. Ta je však již dnes zastaralá a v současném řešení je použita verze Xenu 4.1. Instalace Xenu se skládá ze dvou hlavních debianovských balíčků – *xen-hypervisor*, což je „jádro“ Xenu a balíček *xen-utils*, který je odpovědný za správu a management virtuálních strojů v Xenu. Tyto dva balíčky obsahují závislosti na další balíčky (aplikace), které musí být v systému nainstalovány pro správný chod Xenu. Pro zorientování se v těchto závislostech a instalaci potřebných balíčků nám zajistí např. v Linuxu dobře známá aplikace Advanced Package Tool (apt), případně nástroj Debian Package Management System (dpkg). Jmenuji zde některé důležité závislostní balíčky jako je *e2fslibs*, pro přímý přístup do ext2 souborových systémů, balíčky *iproute* a *bridge-utils*, o kterých se zmíním později a pak balíčky, které jsou vázány přímo na Xen – *libxenstore*, *xen-utils-common*, *libc6-xen* a další. Kromě těchto závislostních balíčků je třeba nainstalovat balíček *xen-linux-system*, což je upravené systémové jádro s podporou virtualizace a balíček *linux-image-2.6-xen*, který je na předešlém balíčku závislý a představuje linuxový kernel spustitelný pouze v prostředí hypervisoru (hostitelského OS). Tento kernel podporuje jak operace privilegované Domény 0, tak i neprivilegovaných domén U.

Všechny výše vypsané balíčky jsou dohledatelné a stažitelné na oficiálních stránkách podpory Debianu [17]. Důvod, proč jsem nepoužil instalaci Xenu z oficiálních domovských stránek Xenu [4], ale ze stránek podpory Debianu je následující. Přesto, že jsem u oficiální instalace Xenu postupoval přesně krok po kroku dle uživatelské dokumentace [5], tak se mi instalaci nepodařilo úspěšně rozchodit a Xen stále vykazoval chyby. Druhá možnost byla stáhnout si zdrojové kódy Xenu, které se následně zkompilují. I v tomto případě jsem postupoval podle oficiální uživatelské dokumentace, ale opět neúspěšně. Vysvětlení může vězet v tom, že oficiální instalace Xenu je podporována napříč všemi distribucemi Linuxu, kdežto instalace ze stránek podpory Debianu je odladěná pro distribuci Debian.

4.1.3 Instalace ostatních potřebných komponent

Kromě samotného Xenu budeme pro naše virtualizační řešení potřebovat další aplikace, zejména síťového charakteru. První z nich je debianovský balíček *bridge-utils*. Tento balíček obsahuje utility pro konfiguraci Linux Ethernet bridge. Linux Ethernet bridge může vytvářet vícenásobné spojení několika ethernetových zařízení. Takové spojení je plně transparentní – hosté

připojení k jednomu ethernetovému zařízení vidí připojené hosty u ostatních ethernetových zařízení. Je to jedno z možných síťových zapojení Xebu a bude podrobněji probráno později v této kapitole. Dalším balíčkem je *vlan*, který umožňuje vyvářet a mazat VLAN sítě na vlastním ethernetovém zařízení. Neobejdeme se jistě bez balíčku *iproute*, což je kolekce utilit pro práci v síti a kontroly toku dat na síti. Posledním balíčkem z této rodiny je *xinetd*, tzv. super-server, který řídí tok síťových služeb na Internetu. Xinetd funguje tak, že místo několika démonů jednotlivých služeb, trvale běží a na příchozí požadavky čeká jediný proces super-serveru, který teprve dle potřeby vyvolá pro obsloužení konkrétního požadavku server příslušné služby.

Další debianovský balíček, bez kterého se neobejdeme, se nazývá *debootstrap*. Nástroj deboostrap dokáže vytvořit Linux systém bez nutnosti instalátorů dpkg či apt. Debootstrap nejprve stáhne z internetového úložiště všechny debianovské balíčky pro danou verzi Linuxu a všechny je poté nainstaluje do klasické linuxové adresářové struktury, která tvoří souborový systém. Takto vytvořený souborový systém poslouží pro vytvoření výchozího hostovaného operačního systému, který bude ve Virtlabu použit, a který se načte na všech virtuálních instancích po začátku rezervace. Pokud bychom nechtěli vytvářet kompletní hostovaný systém od začátku a ručně, tak můžeme použít aplikaci *xen-tools*, která dokáže snadno a rychle vytvořit plně funkční nakonfigurovaný hostovaný systém (domU) a ten potom dále modifikovat či kopírovat.

Poslední aplikace, kterou pro naši práci potřebujeme je *LVM*, o které již byla řeč v minulé kapitole a která je ke stažení na stránkách podpory Debianu ve formě stejnojmenného balíčku.

4.2 Konfigurace Xenu

Po instalaci Xenu se Xen nachází v defaultním stavu, který však není příliš použitelný. Je nutno Xen nakonfigurovat do stavu, jaký vyžadujeme.

4.2.1 Konfigurace GRUBu

Po instalaci a rebootování systému naběhne zavaděč systému zvaný GRUB, kde nyní přibyla jedna nová položka – možnost spustit Doménu 0, tj. hostující operační systém Xenu. Do domény 0 by se uživatel měl nabootovat jako do klasického Linuxu včetně přihlášení.

Po přihlášení si můžeme ověřit, že Xen opravdu provedl změnu v konfiguraci GRUBu. V adresářové složce */boot/grup* si vyhledáme soubor *menu.lst*, což je vstupní konfigurační soubor pro linuxový zavaděč operačního systému zvaný GRUB, jehož úkolem je nakopírovat jádro Linuxu do paměti a spustit ho. Jelikož Xen využívá paravirtualizaci, která vyžaduje speciálně upravený hostující systém, je jasné, že systém nemůže být nastartován klasickým způsobem ale s jistými změnami. Tyto změny si můžeme prohlédnout právě v souboru */boot/grup/menu.lst*. Xen během instalace přidá do konfiguračnho souboru řádky podobné těmto:

```
title           Xen 3.2-1-i386 / Debian GNU/Linux, kernel 2.6.26-2-xen-686
root            (hd1,0)
kernel          /boot/xen-3.2-1-i386.gz
module          /boot/vmlinuz-2.6.26-2-xen-686 root=UUID=16bcbab8-806f-4b48-a64e-aba2e7b3c6e8 ro console=tty0
module          /boot/initrd.img-2.6.26-2-xen-686
```

Obrázek 4.1: GRUB - fragment souboru /boot/menu.lst

První řádek obsahuje title text, který se objeví v menu při výběru bootování systému. Další řádek udává, na kterém diskovém oddílu se nachází složka */boot*. Ve třetím řádku je obyčejně napsáno, jaké jádro systému chceme nahrát, v tomto případě se však jedná o samotný Xen. Následující řádek popisuje umístění upraveného kernelu, který by měl Xen při bootování systému nahrát. Paravirtualizace vyžaduje pro svůj běh speciálně upravený hostující systém, což je také důvod použití Xenem upraveného kernelu. Poslední pátý řádek pak nahraje ramdisk, což je schéma pro nahrání dočasného souborového systému do paměti během bootování kernelu.

4.2.2 Konfigurace Domény 0 (dom0)

Za systémový management je v Xenu odpovědný démon Xend. Je to defacto hlavní řídící prvek Xenu a musí být spuštený, aby bylo možno pracovat s virtuálními stroji. Xend může být spuštěn a ovládán pouze s rootovskými právy. Skript Xend démona je uložen v adresáři */etc/init.d*. Xend můžeme ovládat pomocí příkazů *xend start*, *xen stop* či můžeme příkaz zavolat s kompletní cestou */etc/init.d/xend start*, */etc/init.d/xend stop*.

Druhý démon, který nám Xen poskytuje se nazývá *xendomains*. Tento démon pokud je aktivován (*/etc/init.d/xendomains start*, */etc/init.d/xendoimans stop*), se stará o automatický start virtuálních stanic (domU) během bootování hostujícího operačního systému (dom0) a také o automatické vypnutí virtuálních stanic při odstavení hostujícího systému. Aby se určitý virtuální stroj automaticky spouštěl a vypínal souběžně s Doménou 0, je nutno vložit do složky */etc/xen/auto* konfigurační soubor daného virtuálního stroje. Konfigurační soubor virtuálního stroje bude vysvětlen později v této kapitole.

V případě nesrovnalostí či defektů nabízí Xen funkcionalitu logování. Logovací data můžeme najít v souboru */var/log/xen/xend.log* (nebo také */var/log/xen/xend-debug.log*).

Konfigurační soubor xend-config.sxp

Velmi důležitou roli hraje konfigurační soubor */etc/xen/xend-config.sxp*, který se načítá při bootování Domény 0 a určuje chování Xenu. Pro lepší přehled, co vše lze v tomto souboru nakonfigurovat, uvádím níže seznam některých jeho vlastností:

- *logfile* – umístění logovacího souboru
- *loglevel* – úroveň logování (INFO, WARNING, ERROR, CRITICAL)

- *network-script* – Název skriptu ve složce */etc/xen/scripts*, který bude použit pro nastavení síťového prostředí. Konkrétní nastavení hodnot této proměnné bude předmětem podkapitoly Síťová konfigurace Xenu.
- *vif-script* – Název skriptu ve složce */etc/xen/scripts*, který bude použit pro nastavení virtuálních rozhraní hostovaných domén (domU). Nastavení tohoto parametru je nepřímo závislé na parametru network-script a nastavení hodnot této proměnné bude rovněž probráno v podkapitole Síťová konfigurace Xenu.
- *dom0-min-mem* – Tento parametr určuje, jaké mimimální množství operační paměti bude rezervováno pro Doménu 0.
- *dom0-cpus* – Specifikuje počet procesorů (procesorových jader), které Doména 0 může používat pro běh virtuálních strojů. Jestliže tato proměnná má hodnotu 0, tak všechny dostupné procesory budou využívány Doménou 0.

Xen management (XM)

Xen management (XM) je hlavní rozhraní pro správu a management virtuálních strojů (domU) v Xenu. Jedná se o command line tool. Tato aplikace může být například použita k vytvoření, zastavení či smazání virtuální instance. Dále je tento nástroj schopen vypsat seznam všech aktuálně existujících virtuálních instancí, včetně toho v jakém stavu se nacházejí. Nástroj xm má jasně danou strukturu příkazu:

`xm operace_xenu id_domeny [dalsi_parametry]`.

```
Common 'xm' commands:

console             Attach to <Domain>'s console.
create              Create a domain based on <ConfigFile>.
new                 Adds a domain to Xend domain management
delete              Remove a domain from Xend domain management.
destroy             Terminate a domain immediately.
dump-core           Dump core for a specific domain.
help                Display this message.
list                List information about all/some domains.
mem-set             Set the current memory usage for a domain.
migrate             Migrate a domain to another machine.
pause               Pause execution of a domain.
reboot              Reboot a domain.
restore             Restore a domain from a saved state.
resume              Resume a Xend managed domain
save                Save a domain state to restore later.
shell               Launch an interactive shell.
shutdown            Shutdown a domain.
start               Start a Xend managed domain
suspend             Suspend a Xend managed domain
top                 Monitor a host and the domains in real time.
unpause             Unpause a paused domain.
uptime              Print uptime for all/some domains.
vcpu-set            Set the number of active VCPUs for allowed for
                    the domain.
```

Obrázek 4.2: Možnosti nástroje XM

Jak lze vyčíst z předešlého obrázku, tak např. příkazem `xm console název_domény` se můžeme připojit na konzoli virtuální instance a začít tak pracovat na samotné virtuální instanci. Příkazem `create` vytvoříme novou virtuální instanci, příkazem `delete` ji smažeme. Příkazy `destroy`, `reboot`, `pause` a `unpause` slouží k ovládání běhu virtuální instance. Příkaz `save` a `restore` slouží k ukládání a načítání virtuálních strojů. Pokud požadujeme seznam všech existujících virtuálních instancí, použijeme příkaz `list`. V této chvíli nám příkaz `xm list` vypíše pouze Doménu 0, žádná virtuální instance zatím nebyla vytvořena.

```
debian1:~# xm list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  3408     4     r-----    144.4
```

Obrázek 4.3: Hostující Doména 0

4.2.3 Konfigurace LVM

V minulé kapitole byl představen systém logických disků LVM a popsána funkcionalita této technologie, která umožňuje napříč fyzickými disky vytvářet vrstvu logických disků, se kterými je poté možno pracovat jako s fyzickými disky. Tato technologie nás zajímá hlavně z hlediska možnosti vytvářet nad takto vytvořenými logickými disky jejich obrazy (snapshoty), což bude využito v systému Virtlab. Postup vytvoření logického úložiště je následující:

1) Nejprve je potřeba inicializovat pevný disk, který bude pro LVM použit, tzn. vytvoříme pro LVM fyzický oddíl (physical volume – PV). Fyzické oddíly je možno vytvářet buď na celém disku nebo na oddíle. Ve výpisu příkazu můžeme vidět, že jsme pro LVM vytvořili dva fyzické oddíly z diskových oddílů *sda7* a *sda8*.

```
pvcreate /dev/sda7
pvcreate /dev/sda8
```

2) V dalším kroce je nutné fyzické oddíly vytvořené LVM spojit do jednoho logického celku, do tzv. skupiny oddílů (volume group – VG). Toho dosáhneme příkazem `vgcreate nazev_skupiny seznam_oddilu`.

```
vgcreate vg0 sda7 sda8
```

3) Z vytvořené skupiny oddílů nyní vyseparujeme logický oddíl (logical volume – LV), který již představuje místo, na kterém lze vytvořit systém souborů a který poslouží jako disková jednotka pro souborový systém virtuálního stroje. O vytvoření logického oddílu se stará příkaz `lvcreate`, kde parameter `name` je název logického oddílu, `size` představuje

velikost logického oddílu a na závěr je povinný parametr název volume group, odkud se také vezme potřebný prostor pro vytvoření logického oddílu.

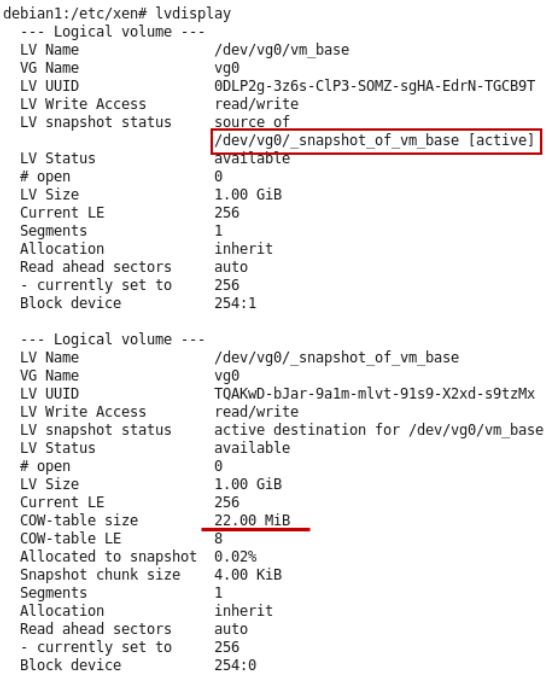
```
lvcreate --name vm_base --size 1G vg0
```

```
debian1:/etc/xen# lvdisplay
  --- Logical volume ---
  LV Name                /dev/vg0/vm_base
  VG Name                vg0
  LV UUID                0DLP2g-3z6s-ClP3-SOMZ-sgHA-EdrN-TGCB9T
  LV Write Access        read/write
  LV Status              available
  # open                 0
  LV Size                1.00 GiB
  Current LE             256
  Segments               1
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           254:1
```

Obrázek 4.4: Vytvořený logický oddíl vm_base

4) Z logického oddílu vytvořeného v LVM je velice snadné udělat snapshot. Postará se o to opět příkaz `lvcreate`, tentokrát však navíc s parametrem `snapshot` a se jménem logického oddílu, ze kterého snapshot vychází. Syntaxe příkazu je následující:

```
lvcreate --snapshot --name _snapshot_of_vm_base --size 20M \
/dev/vg0/vm_base
```



Obrázek 4.5: Ukázka LVM snapshotu

Jak si můžeme na obrázku všimnout, defaultní logický oddíl *vm_base* nyní obsahuje referenci na vytvořený snapshot *_snapshot_of_vm_base*, který je ve výpisu uveden níže, právě pod logickým oddílem *vm_base*.

Při vytváření snapshotu jsme zadávali i parametr pro maximální velikost snapshotu. Z příkazu `lvdisplay` můžeme vyčíst, že snapshot může mít maximální velikost 22 MB. Na první pohled, se to může zdá velice málo, ale po mnoha praktických pokusech jsem došel k závěru, že tato limitující velikost je zcela dostačující. Pokud by tato kapacita byla přesto nedostačující (student provedl změny typu, které jsou náročné na diskový prostor), LVM umožňuje kapacitu snapshotu dodatečně navýšit a to i v případě, že snapshot je již naplněn daty. Tato funkcionalita bude ve správě virtualizovaných stanic využita, to znamená, když systém zjistí, že snapshot je již téměř zaplněn, systém automaticky navýší kapacitu daného snapshotu o určitou jednotku velikosti, ale pouze za předpokladu, že snapshot jiř nepřesáhnul maximální povolenenou velikost. Maximální možnou velikost snapshotu a jednotku navýšení kapacity definuje administrátor Virtlabu. Tyto data jsou následně uložena na virtuálním serveru, kde se nachází samotný XEN.

LVM funkcionalita vytváření snapshotů je klíčovým prvkem k tomu, aby si studenti mohli masově ukládat své rozpracované virtuální instance bez plýtvání diskového prostoru.

4.3 Vytvoření virtuální instance v Xenu (domU)

Jak již bylo v minulých kapitolách napsáno, Xen je paravirtualizační nástroj, který pro svůj běh vyžaduje speciálně upravený hostující systém (dom0). U hostovaného systému (domU) je

situace obdobná, i zde Xen vyžaduje, aby byl hostovaný systém speciálně upraven takovým způsobem, že hostovaná doména domU nepředává své požadavky fyzické vrstvě, ale hostující doméně dom0, která tyto požadavky provede a výsledek následně vrátí hostované doméně domU.

Abychom v Xenu mohli úspěšně spustit virtuální instanci, musíme takovým hostovaným systémem disponovat. Je několik způsobů, jak takový systém získat. Samozřejmě na Internetu lze získat již předpřipravené hostované systémy, ale ty jsou již většinou zpoplatněné, takže této možnosti se vyhneme. Další možností je vytvořit si takový systém od začátku (*angl. from scratch*), což je popsáno v následující podkapitole.

4.3.1 Vytvoření virtuální instance krok za krokem

1) Nejprve je nutné vytvořit diskové úložiště, které bude moci virtuální stroj využívat. Možností máme několik. Pro naši první virtuální instanci použijeme nejrychlejší možné řešení, tj. v Linuxu dobře známa utilita `dd`, která na disku vytvoří blok dat (image) požadované velikosti, který poslouží jako úložiště pro virtuální instanci. Bloky si vytvoříme dva – jeden pro souborový systém virtuálního stroje, druhý jako swapovací oblast. Syntaxe příkazu je následující:

```
dd if=/dev/zero of=/xen/domU.img bs=1k seek=1024k count=1
dd if=/dev/zero of=/xen/domU_swp.img bs=1k seek=256k count=1
```

Tyto příkazy vytvoří dva bloky dat (dva soubory), jeden o velikosti 1024 MB a druhý 256 MB. První je určen pro souborový systém virtuálního stroje, druhý jako swapovací oblast. Tyto soubory mají na počátku nulovou velikost, jelikož diskové místo je pro ně alokováno, teprve až je třeba.

2) Nyní, když máme připravené bloky dat, je třeba na jednom z nich vytvořit klasický linuxový souborový systém, v tomto případě ext3, na druhém bloku pak swapovací oblast.

```
mkfs.ext3 /xen/domU.img
mkswap /xen/domU_swp.img
```

3) Následující série příkazů nejprve vytvoří adresář */mnt/debian_package*, na který se následně připojí námi vytvořený souborový systém */xen/domU.img*. Tímto příkazem jsme učinili soubor */xen/domU.img* přístupný jako klasické blokové zařízení (hardisk). V posledním kroce si pomocí nástroje debootstrap stáhneme z internetového úložiště distribuci Linuxu, která bude nahrána do zpřístupněného souborového systému */xen/domU.img*.

```
mkdir /mnt/debian_package
```

```
mount –o loop /xen/domU.img /mnt/debian_package
debootstrap –arch i386i lenny /mnt/debian_package \
http://archive.debian.org/debian-archive/debian/
```

Po dokončení operace si můžeme všimnout, že ve zpřístupněné složce */mnt/debian_package* se vytvořila typická linuxová adresářová struktura. Všechny debianovské zdrojové balíčky, které deboostrap stáhnul si můžeme prohlédnout ve složce */mnt/debian_package/var/cache/apt/archives*.

4) Nově vytvořenému linuxovému systému však chybí jádro, to zkopírujeme z Domény 0. Hostovaná doména, tak dostane speciálně upravené jádro k virtualizaci. Konkrétní verze kernelu se odvíjí od toho, jakou verzi Linuxu používáme.

```
cp –dpR /lib/modules/2.6.26-2-xen-686 \
/mnt/debian_package/lib/modules
```

5) V následujícím kroce si pomocí utility *chroot* nastavíme nový rootovský adresář, který bude odkazovat na nově vytvořený souborový systém */mnt/debian_package*. Poté nastavíme nové systémové jméno (hostname), zde už však nezadáváme kompletní cestu včetně */mnt* adresářem, ale pouze samotnou cestu k *hostname* souboru díky tomu, že rootovský adresář byl změněn.

```
chroot /mnt/debian_package
echo "domU" > /etc/hostname
```

6) Soubor */etc/fstab* je v linuxovém světě dobře známý, obsahuje tabulku souborových systémů. V našem případě však tento soubor není nakonfigurován. Aby systém věděl, jaké svazky se mají při bootování systému připojit, je třeba tento soubor vhodně upravit, viz. printscreen níže:

```
# Begin /etc/fstab
# <file system> <mount-point>   <type>   <options>                    <dump> <pass>
/dev/sda1          /            ext3      defaults,errors=remount-ro   0     0
/dev/sda2          swap         swap      sw                           0     0
proc               /proc        proc      defaults                     0     0
# End /etc/fstab
```

Obrázek 4.6: Konfigurace souboru /etc/fstab

7) V tomto bodě je tvorba nového hostovaného operačního systému téměř hotova. Chybí už v podstatě pouze sítové nastavení samotné virtuální instance, to však v této chvíli přeskočíme. Příkazem `CTRL + D` se vrátíme zpět do původního rootovského adresáře a odpojíme připojené zařízení */mnt/debian_package*.

```
CTRL + D
chroot /mnt/debian_package
```

8) Posledním nutným krokem před samotným spuštěním virtuální instance je vytvoření konfiguračního souboru, kde jsou uloženy veškeré konfigurační informace o dané virtuální instanci. Tento soubor má příponu *cfg*. Popis možných konfiguračních parametrů je předmětem následující podkapitoly, všimněte si však názvu domény `domU`. Pro práci s hostovanou doménou se bude používat toto jméno. Níže je ukázka, jak může konfigurační soubor pro danou virtuální instanci vypadat.

```
kernel = "/boot/vmlinuz-2.6.26-2-xen-686"
ramdisk = "/boot/initrd.img-2.6.26-2-xen-686"
memory = 128
name = "domU"
disk = ['file:/xen/domU.img,sda1,w','file:/xen/domU_swp.img,sda2,w']
root = "/dev/sda1 ro"
vif = [ 'ip=192.168.1.101,mac=00:16:3E:71:9E:49' ]
extra = "console=hvc0 xencons=tty clocksource=jiffies"
on_poweroff = 'destroy'
on_reboot   = 'restart'
on_crash    = 'restart'
```

Obrázek 4.7: Konfigurační soubor domU.cfg

9) Nyní je vše připraveno pro první spuštění virtuálního stroje, tzn. nejprve pomocí nástroje `xm` danou virtuální instanci vytvoříme a následně se na ní z Domény 0 přes konzoli připojíme. Do systému se přihlásíme jako `root`, heslo použijeme stejné, pokud jej systém vyžaduje. Síťové prostředí zatím nebylo nakonfigurováno, takže virtuální instance je pro okolní svět neviditelná a ani ona nemá žádné vnější spojení se světem, pouze s Doménou 0 přes konzoli.

```
xm create /xen/domU.cfg
xm console domU
root / root
```

```
debian1:~# xm list
Name                                        ID   Mem VCPUs      State   Time(s)
Domain-0                                     0  3408     4     r-----    141.7
domU                                         3   128     1     -b----      2.4
```

Obrázek 4.8: Běžící virtuální instance domU

10) Jakmile se z konzole virtuálního stroje chceme odhlásit, ale chceme nechat virtuální stroj běžet dál, zadáme klávesovou zkratku `CTRL + ]`. Pokud požadujeme běh virtuální instance ukončit, použijeme k tomu xen management tool.

```
CTRL + ]
xm shutdown domU
```

4.3.2 Konfigurační soubor hostované Domény U

Každý virtuální stroj v Xenu (domU) má svůj vlastní konfigurační soubor, který obsahuje sadu specifických parametrů pro danou hostovanou Doménu U, které určují, jak se daná doména bude chovat. Níže uvádím přehled hlavních vlastností konfiguračního souboru pro hostovanou doménu běžícím v Xenu. Parametry *kernel*, *name*, *disk* a *root* jsou povinné, ostatní parametry jsou volitelné.

- *kernel* – image kernelu, jaký bude hostovaná instance využívat
- *ramdisk* – image ramdisku, který se použije při botování kernelu
- *memory* – velikost operační paměti vyhrazená pro virtuální instanci
- *name* – jméno virtuální instance, musí být unikátní, využívá se např. při operacích s nástrojem xm
- *disk* – tento parametr obsahuje disková zařízení, které bude hostovaná doména využívat, diskové zařízení může být dvou typů:
    - *file* – diskové zařízení zde představuje image souborového systému, stejný typ zařízení byl použit v předchozí kapitole, kdy jsme vytvářeli virtuální instanci od začátku, hlavní nevýhoda tohoto typu úložiště spočívá v jeho nedostatečné rychlosti oproti klasickému fyzickému zařízení
    - *phy* – fyzické zařízení může být několika typů, může se například jednat o klasický pevný disk (např. `disk = [´phy:hda3,sda1,w´]`, kde hda3 je fyzické zařízení, které bude do virtuální instance exportováno jako zařízení sda1 pro čtení i zápis – parametr ´w´), či se může jednat o disk partition, v tomto případě o LVM partition, která bude využita v diplomové práci (např. `disk = [´phy:vg/lvm1,sda2,w´]`)
- *nfs_root*, *nfs_server* – parametry potřebné pro nastavení přístupu ke vzdálenému úložišti pomocí protokolu Network File System
- *root* – nastavení rootovského adresáře hostované domény
- *vif* – je zkratka od slova virtual interface a jak už název napovídá, jedná se o nastavení síťového rozhraní pro danou virtuální instanci, v tomto parametru může být uloženo hned několik hodnot jako například ip adresa virtuálního stroje, mac adresa či název ethernetového bridge, ke kterému je hostovaná doména připojena
- *extra* – obsahuje speciální parametry, které budou přidány ke kernelu při spouštění virtuální instance

- *on_poweroff* – parametr určuje, jaká operace nastane při odstavení virtuální instance, platné hodnoty tohoto parametru jsou – destroy, restart, preserve, rename-restart
- *on_reboot* – jaká operace nastane při restartu virtuální instance, tento parametr nabývá stejných hodnot jako parametr on_poweroff
- *on_crash* – jaká operace nastane v případě, že virtuální instance zhavaruje, tento parametr nabývá stejných hodnot jako parametr on_poweroff
- *on_xend_start* – určuje, zda se má při startu démona Xend nastartovat také virtuální instance, možné hodnoty – ignore, start
- *on_xend_stop* – parametr určuje chování virtuální instance v případě, že se ukončí démon Xend, možné hodnoty tohoto parametru jsou – ignore, shutdown, suspend

4.3.3 Vytvoření virtuální instance pomocí nástroje xen-tools

Manuální úroces vytváření virtuálních stanic, jak jsme si ukázali v předminulé kapitole, je docela zdlouhavý. Nástroj xen-tools tento problém řeší efektivně tím způsobem, že proces vytváření virtuální instance do velké míry automatizuje. Nástroj xen-tools je tvořen sadou skriptů napsaných v jazyce Perl.

Xen-tools používá konfigurační soubor */etc/xen-tools/xen-tools-conf*, který v sobě ukrývá rozsáhlou sadu parametrů, na základě kterých je virtuální instance vytvořena, kopírována či smazána. Nástroj xen-tools se ovládá pomocí tří příkazů:

- *xen-create-image* – jak už název napovídá, tato utilita vytvoří novou virtuální instanci a poté pro ni nastaví síťové prostředí
- *xen-list-images* – vypíše seznam vytvořených virtuálních instancí společně s jejich síťovým nastavením
- *xen-delete-image* – tento příkaz smaže dříve vytvořenou hostovanou doménu

Každý z výše uvedených třech příkazů obsahuje bohatou manuálovou nápovědu, konfigurační soubor obsahuje četné komentáře ke každému parametru, tudíž této látce se nebudu dále hlouběji věnovat.

4.3.4 Časté chyby při vytváření virtuálních instancí

Device 0 (vif) could not be connected

Celé znění této chyby je – *Error: Device 0 (vif) could not be connected. Could not find bridge, and none was specified*. Tato chyba na nás vyskočí ihned poté, co se pokusíme vytvořit naši první virtuální instanci tím, že zadáme příkaz `xm create config_guest_file`. Důvodem této chyby je, že po instalaci Xenu se nachází konfigurační soubor */etc/xen/xend-config.xsp* v defaultním stavu, který je však nastaven nesprávně. Pro parametr *network-script*,

který určuje síťové nastavení hostované domény, je nastavena hodnota *network-dummy*, která síťové nastavení pro hostovanu doménu ponechává čistě na nás. Zpočátku zde však nic není nastaveno a z tohoto důvodu nám Xen při vytváření virtuální intance vyhodí výše zmíněnou chybu. Rychlá náprava spočívá v zakomentování hodnoty *network-dummy* a odkomentování hodnoty *network_brigde* pro parametr *network-script*. Aby tato změna nabyla účinku je nutno restartovat démona Xend.

Tato změna nám pouze zaručí to, že od této chvíle lze vytvářet nové virtuální instance, ne že bude správně nastaveno síťové prostředí pro hostovanou doménu. V této diplomové práci si síťové prostředí nastavíme sami, jelikož ostatní Xenem nabízená síťová nastavení jsou pro nás nevyhovující, to znamená, že hodnota parametru *network-script* bude *network-dummy*. Síťová konfigurace je podrobně popsána v následující podkapitole.

Nemožnost přístupu ke konzoli virtuální instance

Poté, co jsme úspěšně vytvořili první virtuální instanci v Xenu, si nyní chceme ověřit, že vše proběhlo správně. Příkazem `xm console název_domény` se připojíme na konzoli dané virtuální instance. Po zadání tohoto příkazu se úspěšně spustí bootování souborového systému virtuální instance, to však vždy zamrzne bez další možnosti přihlásit se na virtuální instanci. Důvod je ten, že Xen používá konzoli zvanou *hvc0*, kdežto v souborovém systému virtuální instance je nastavena konzole *tty*.

Tuto chybu můžeme opravit dvěmi způsoby. Do konfiguračního souboru cfg pro danou hostovanou doménu přidáme následující řádek *extra = "console=hvc0 xencons=tty"*. Tyto dva parametry budou předány jádru kernelu při spouštění virtuální instance a zajistí, že hostující doména (dom0) a hostovaná doména (domU) budou používat stejný typ konzole. Druhý způsob spočívá v tom, že si existující diskové úložiště se souborovým systémem dané vituální instance přimontujeme a pozměníme soubor */etc/inittab*, který určuje chování systému při bootování. V tomto souboru si vyhledáme řádek *1:2345:respawn:/sbin/getty 38400 tty1*, kde hodnotu *tty1* přepíšeme na *hvc0*.

clocksource/0: Time went backwards

Pravděpodobně se jedná o nejslavnější chybu Xenu. Chyba se projevuje tím způsobem, že konzole virtuální instance je soustavně „zasypávána" zprávami typu *clocksource/0: Time went backwards: delta=-5802595381342 shadow=440273248432 offset=11586903* apod. Toto neustálé zasypávání prakticky znemožňuje jakoukoliv práci s virtuální instancí a většinou nakonec vede k úplnému zatuhnutí virtuální instance.

Příčina této chyby tkví v tom, že virtuální instance používý časový signál Xenu, namísto toho, aby používala vlastní. Náprava této chyby je nasnadě. Do konfiguračního souboru dané virtuální instance přidáme do parametru *extra* další dvojici dat (*clocksource=jiffies*), která při bootování kernelu virtuální instance přenastaví časový signál z hodnoty *xen* na hodnotu *jiffies*. Jiffies je globální proměná v linuxovém kernelu, která inkrementálně vzrůstá o 1 a uchovává

hodnotu, která říká, kolik kmitů uplynulo od nabootování systému. Tato proměnná je použita pro rozličné časovací funkce uvnitř kernelu.

4.4 Síťová konfigurace Xenu

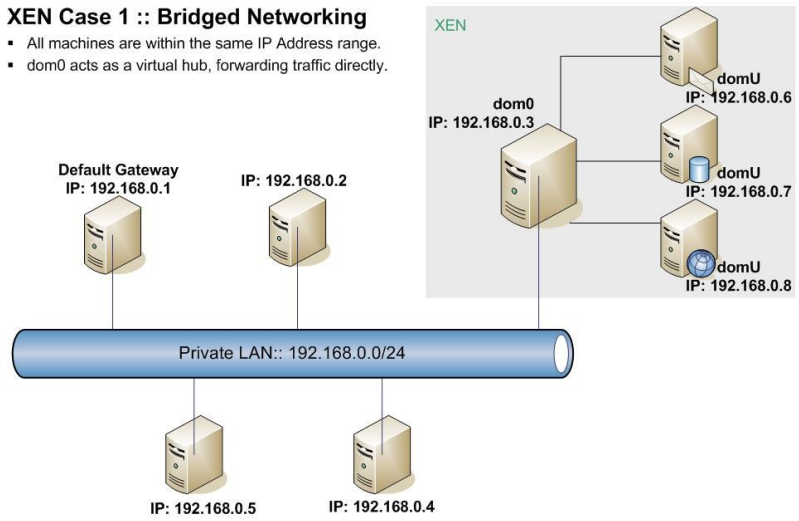
Xen je zhlediska možnosti síťové konfigurace velice variabilní a nabízí mnoho možností, jak nakonfigurovat síťové prostředí na míru a v závislosti na daném prostředí. Síťová konfigurace Xenu může být postavena buď na již existujícím síťovém prostředí či na zcela nově vytvořené privátní síti, která jasně vymezuje virtuální síťový prostor. Xen má 4 hlavní defaultní konfigurace, které lze nastavit v konfiguračním souboru */etc/xen/xend-config.sxp*. Typicky lze nastavit v čase pouze jednu konfiguraci, ostatní možnosti musí být zakomentovány.

Bridge networking

První možností síťové konfigurace je Bridge networking. Pokud se pro tuto konfiguraci rozhodneme, je třeba v konfiguračním souboru Xenu (*/etc/xen/xend-config.sxp*) nastavit následující parametry:

```
(network-script network-bridge)
(vif-script vif-bridge)
```

Bridge networking je nejsnadnější a také nejrychlejší způsob, jak nakonfigurovat síťové prostředí v Xenu. Virtuálním strojům umožňuje jednoduše se připojit do existující síťové struktury za použití jejich virtuálních ethernetových karet. Toto síťové nastavení zpravidla využijeme v situacích, kdy potřebujeme připojit virtuální stroj k již existující síti používající protokol DHCP či statické IP adresy nebo v případě, že požadujeme, aby virtuální stroj byl plně viditelný a přístupný na existující síťové infrastruktuře, dovolující síťový provoz (*ang. network traffic*) v obou směrech. Všechny virtuální stroje v tomto nastavení používají stejný rozsah IP adres. Doména 0 zde hraje úlohu virtálního hubu, přeposílajícího packety od virtuálních strojů do ostatních částí sítě a naopak.

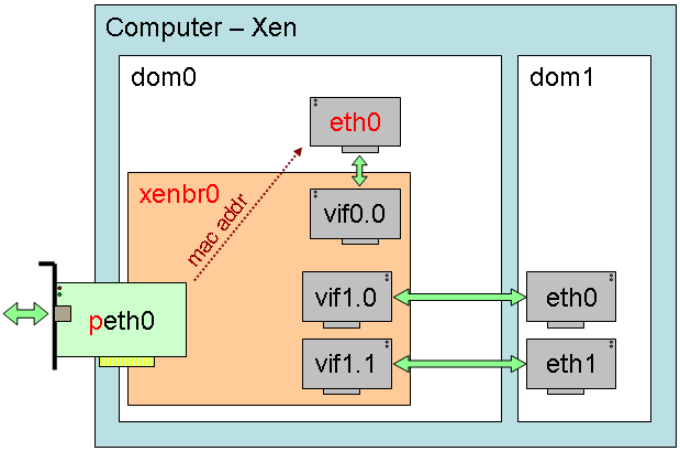


Obrázek 4.9: Bridge networking

Pro lepší představu, jak funguje komunikace mezi Doménou 0 a hostovanou doménou (domU) , zde uvádím detailnější pohled síťového nastavení pro bridge networking.

Xen skript `network-bridge` nejprve vytvoří nový ethernetový bridge nazvaný *xenbr0*. Poté shodí fyzické síťovérozhraní *eth0*, které přejmenuje na *peth0*. Virtuální rozhraní Domény 0, zvané *veth0*, je přejmenováno na *eth0* (červené písmo). IP adresa a MAC adresa fyzického rozhraní (nyní *peth0*) je zkopírována do virtuálního rozhraní *eth0*. V dalším kroce je fyzické rozhraní *peth0* a virtuální rozhraní *vif0.0* přidáno do bridge *xenbr0*. Nakonec jsou rozhraní *peth0*, *eth0* a *vif0.0* včetně bridge *xenbr0* aktivována. Ihned po dokončení skriptu `network-bridge` následuje ve své činnosti skript `vif-bridge`. Ten připojí virtuální rozhraní, v tomto případě *vif1.0* a *vif1.1*, do bridge *xenbr0* a poté virtuální rozhraní aktivuje. Tato virtuální rozhraní, umístěné v Doméně 0, jsou připojeny k rozhraním virtuálních strojů (*eth0*, *eth1*) a probíhá zde tak síťová komunikace mezi Doménou 0 a ostatními hostujícími doménami.

Po detailním popisu funkčnosti Xen skriptů pro aktivaci síťové konfigurace s ethernetovým bridgem se nyní ještě zmíním o cestě síťového packetu v této konfiguraci. Jakmile packet přistane na fyzickém rozhraní síťové karty, je zpracován ethernetovým rozhraním Domény 0 a objeví se na rozhraní *peth0*. Rozhraní *peth0* je svázáno s ethernetvým bridgem, takže packet je přesunut do tohoto bridge. Tento krok se odehrává na druhé vrstvě referenčního ISO/OSI modelu, takže žádná IP adresa pro přesun takového packetu není použita. Nyní bridge distribuuje přijaté packety do jednotlivých virtuálních instancí (domU) jako klasický switch, tzn. na základě MAC adres. Tento packet je tak předán jednomu z virtulních rozhraní (*vif1.0* a *vif 1.1*), které jsou připojeny k bridgy. Takové rozhraní poté packet odešle do cílové stanice – virtuální instance. [7]

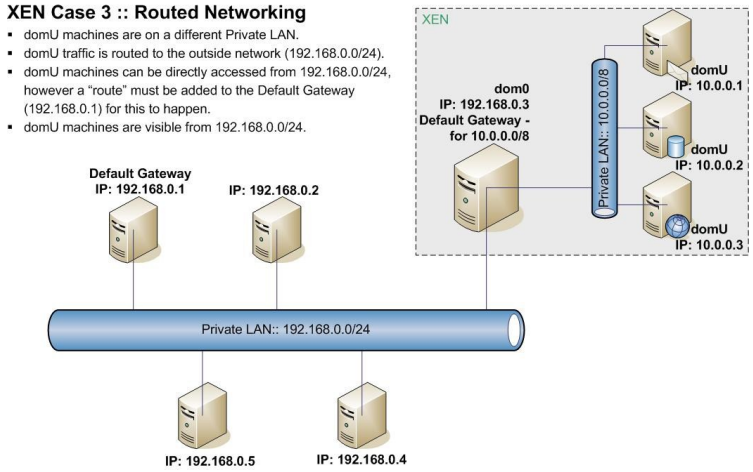


Obrázek 4.10: Bridge networking - vnitřní architektura

Routed networking

Toto síťové nastavení Xenu se rovněž aktivuje v konfiguračním souboru */etc/xen/xend-config.sxp* pomocí dvou parametrů:

```
(network-script network-route)
(vif-script vif-route)
```



Obrázek 4.11: Routed networking

Při této konfiguraci se nacházejí Domény U na vytvořené privátní LAN síti, která je tak oddělená od zbytku sítě. Mezi hostující doménou (dom0) a každou hostovanou doménou (domU) se vytvoří tzv. point-to-point odkazy, což znamená, že cesty v síti (*angl. routes*) ke každé Doméně U jsou přidány do routovací tabulky (*angl. routing table*) hostující Domény 0, z čehož vyplívá, že

hostované domény musí mít statickou IP adresu. DHCP v tomto zapojení tudíž nemůže být použito.

Routed networking with NAT

Pro aktivaci této síťové konfigurace je třeba v souboru */etc/xen/xend-config.sxp* nastavit dva parametry:

```
(network-script network-nat)
(vif-script vif-nat)
```

NAT networking se bežně používá, když požadujeme pro Doménu 0 (dom0) a virtuální intance (domU) vytvořit privátní lokální síť. Síťový provoz přicházející z virtuálních strojů je přesměrován do vnější sítě skrz NAT (Network Address Translation). Virtuální stroje jsou v tomto zapojení ukyty a chráněny před vnější sítí. V této konfiguraci bude Doména 0 automaticky vykonávat veškerou funkci NAT zařízení.

Vlastní síťová konfigurace

Pokud nám nevyhovuje žádná z přednastavených sítových rozhraní Xenu, Xen nám umožňuje vytvořit si zcela vlastní síťové nastavení dle našich potřeb. Toho dosáhneme tím, že v konfiguračním souboru */etc/xen/xend-config.sxp* nastavíme pro parametr `network-script` hodnotu `network-dummy`, která určuje, že Xen ponechá síťovou konfiguraci pouze na nás, to znamená, žádné nastavení ze strany Xenu není provedeno. Jelikož nám pro řešení Virtlabu nevyhovuje žádné z přednastavených síťových konfigurací Xenu, použijeme právě tuto variantu nastavení. Konečná síťová konfigurace pro řešení Virtlabu bude následující:

```
(network-script network-dummy)
```

Výsledné síťové nastavení bude podobné jako v případě Bridge networking s několika menšími odlištnostmi. Stejně jako skript `network-bridge`, vytvoří skript `network-dummy` nejprve ethernetový bridge *br0*. Poté přejmenuje fyzické síťové rozhraní *eth0* na *peth0*. Virtuální rozhraní Domény 0, zvané *veth0*, je přejmenováno na *eth0*. Virtuálnímu rozhraní *veth0* (nyní *eth0*) je manuálně přidělena MAC adresa. IP adresa fyzického rozhraní (nyní *peth0*) je přenesena do virtuálního rozhraní *eth0*, kde je poté zároveň nastavena adresa pro broadcast. Nastavení fyzického rozhraní *peth0* je zahozeno. V dalším kroce je fyzické rozhraní *peth0* a virtuální rozhraní *vif0.0* přidáno do bridge *br0*. Nakonec jsou rozhraní *peth0*, *eth0* a *vif0.0* včetně bridge *br0* aktivována.

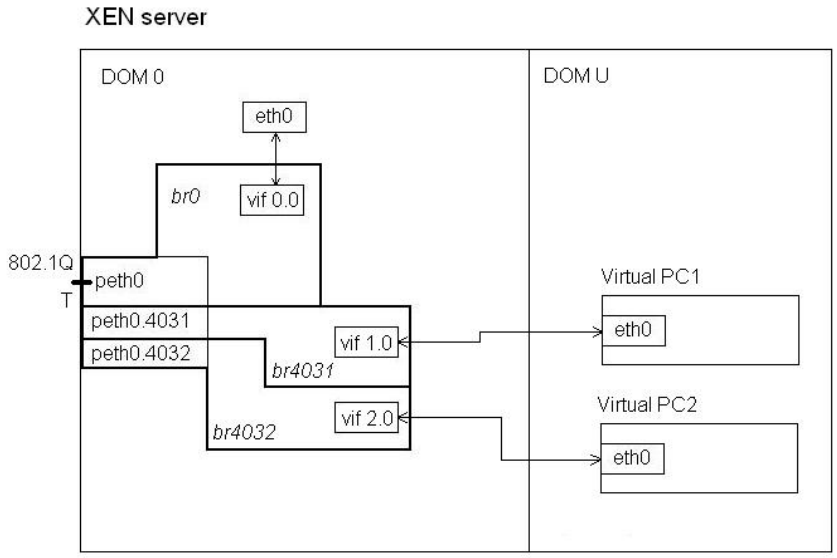
Během vytváření virtuální instance je vytvořen bridge s unikátním názvem (např. *br4031*). Dále je vytvořeno virtuální „podrozhraní“ (VLAN rozhraní) k rozhraní *peth0* rovněž

s unikátním názvem analogicky k předchozímu případu, tj *peth0.4031*. Následně je rozhraní VLAN *peth0.4031* přidáno do bridge *br4031* a oba tyto síťové prvky jsou aktivovány. Díky tomuto řešení je pro každou virtuální instanci vytvořeno jedno unikátní VLAN rozhraní (podrozhraní *peth0*), které určuje, k jaké virtuální instanci data poputují, když dorazí na fyzické rozhraní *peth0* (sloužící v tomto případě jako trunk linka). Tato technologie se nazývá Q-in-Q (založená na standardu 802.1q) a je blíže zmíněna v třetí kapitole. Vytvoření bridge i rozhraní VLAN vykonává skript *bridge.sh*, který je zmíněn v následující podkapitole.

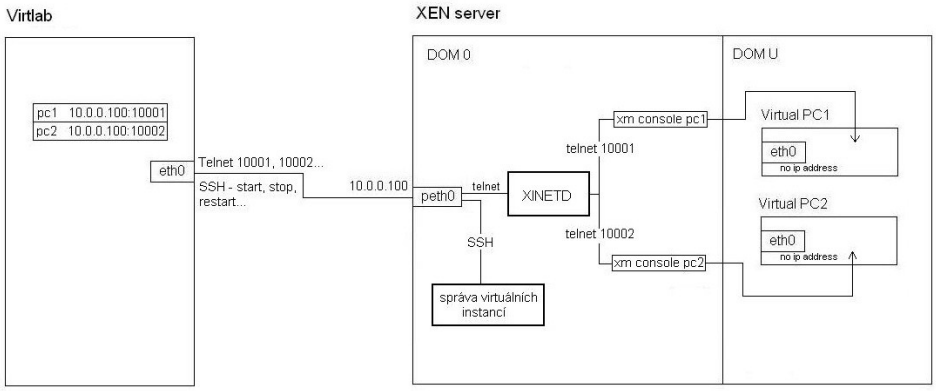
Aby mohl student s danou virtuální instancí začít pracovat, musí se nejprve připojit na její konzoli. Toho docílí s pomocí aplikaci xinetd, zmíněná v podkapitole Instalace ostatních potřebných komponent, která vytvoří xinetd démon pro každou virtulní instanci. Každý takto vytvořený démon obsahuje unikátní číslo portu (např. *10001*), na kterém démon naslouchá a čeká na příchozí spojení. Jakmile student s tímto démonem na určitém portu naváže kontakt (např. aplikace Telnet), démon xinetd spustí konzoli pro danou virtuální instanci. Tato činnost je vykonávána skriptem *xinetd.sh* zmíněným v následující podkapitole.

Všechny až doposud popisované kroky probíhají na Xen serveru. Systém Virtlab komunikuje s Xen serverem na základě studentových požadavků, které jsou Xen serveru předány pomocí SSH protokolu.

Celý předcházející slovní popis síťové architektury Xenu je zachytitelný na následujících dvou diagramech.



Obrázek 4.12: Síťová konfigurace Xenu



Obrázek 4.13: Komunikace Virtlabu s Xen serverem

4.5 Implementace správy virtualizovaných stanic v jazyce Bash

V této kapitole se dostáváme ke stěžejní části diplomové práce. Z analýzy víme, do jakých stavů se během svého životníhoho cyklu může virtuální instance dostat, takže teď tyto poznatky převedeme do programového kódu.

Základní stavební kámen tvoří soubor s názvem **xen.sh**, který funguje jako centrální systém správy virtualizovaných stanic a na základě vstupních parametrů určí, co se bude s danou virtuální instancí dále dít, případně jaká operace bude provedena. Pokud například spustíme tento skriptovací soubor s parametrem *start* a názvem instance virtuálního stroje, tak se vykonají potřebné operace pro spuštění zvolené virtuální instance. Analogicky lze virtuální instanci zastavit, načíst, restartovat, uložit či smazat. Pokud se jakákoliv z výše uvedených operací s virtuální instancí nezdaří a nedoběhne až do konce, je zde zaimplementována funkcionalita, která po sobě „vyčistí“ veškeré dosud provedené akce a tím zajistí, že server, kde probíhá správa virtualizovaných stanic, bude udržován v dobré „kondici“.

Kromě operací svázaných přímo s chodem virtuální instance, skript xen.sh poskytuje také další doprovodné funkce, jako například zjištění aktuálního stavu vybrané virtuální instance (zda je instance běžící či nikoliv), zjištění velikosti zbývajícího volného diskového prostoru pro uložení snapshotů virtuálních instancí, ověření celkové diskové kapacity pro ukládání virtuálních instancí a plus samozřejmě další interní metody nutné pro korektní chod správy virtualizovaných stanic. Skript xen.sh neobsahuje žádný kód potřebný k provedení výše zmíněných operací, pouze metody odkazující se na příslušné skripty, které již požadovanou funkcionalitu provedou. Kompletní výčet všech operací, které je možno provést, lze získat zavoláním tohoto skriptu s parametrem *help*.

Seznam všech virtuálních stanic, které lze ve Virtlabu použít, společně s detailními informacemi potřebnými pro vytvoření takové virtuální instance se nachází v souboru *startup.image.description*. Naproti tomu seznam všech aktuálně běžících virtuálních instancí se nachází v souboru *running.xen.image*. Pokud student ve Virtlabu nastartuje virtuální instanci, pak je záznam o této instanci překopírován ze souboru *startup.image.description* do souboru

running.xen.image. Při zastavení virtuální instance je záznam ze souboru *running.xen.image* vymazán.

Kompletní seznam všech skriptů a souborů, které jsou zakomponovány do správy virtualizovaných stanic, je vypsán v následující části.

Komponenty správy virtualizovaných stanic:

- **xen.sh** – „jádro“ správy virtualizovaných stanic ve Virtlabu
- **common_settings.sh** – skript definující sadu parametrů potřebných pro chod virtualizovaných stanic, jako například maximální disková kapacita pro ukládání snapshotů na jednoho studenta, maximální velikost jednoho snapshotu, jednotka navýšení kapacity diskového prostoru, název výchozího image pro virtuální instance apod.
- **msg.sh** – tento zajišťuje funkcionalitu logování so souboru *log.file*
- **db_inst.sh** – skript zajišťující operace nad soubory *startup.image.description* a *running.xen.image*, v tomto skriptu je zároveň obsažena funkcionalita zpětného „vyčištění“ dat po neúspěšně provedené operaci nad virtuální instancí
- **lvm.sh** – skript sloužící pro prácí s LVM, obsahuje rovněž funkcionalitu navýšení kapacity snapshotu v případě, že jeho velikost je již nedostačující
- **instance.sh** – tento skript vytvoří na základě vstupu z startup.image.description souboru konfigurační cfg soubor, na základě kterého může být následně vytvořena požadovaná virtuální instance
- **instance.conf** – šablona pro vytvoření konfiguračního cfg souboru
- **mac.sh** – skript, který pro každou nově vytvořenou virtuální instanci vytvoří jedinečnou unikátní MAC adresu
- **bridge.sh** – skript zajišťuje síťové nastavení Xenu, tento skript vytvoří pro každou virtuální instanci vlastní bridge a VLAN, přes který se potom virtuální instance připojí k síti
- **xinetd.sh** – skript, který nám umožní připojit se ke konzoli dané virtuální instance
- **xinetd.conf** – šablona, ze které je vytvořen konfigurační soubor pro aplikaci xinetd, která nám umožní se z Virtlabu připojit ke konzoli dané virtuální instance
- **startup.image.description** – soubor definující seznam všech virtuálních stanic, které lze ve Virtlabu použít
- **running.xen.image** – soubor obsahující seznam běžících virtuálních instancí
- **log.file** – logovací soubor, kde jsou zapsány veškeré provedené operace nad virtuálními instancemi

5 Zakomponování do systému Virtlab

Jak už bylo na konci třetí kapitoly zmíněno, Virtlab je webová aplikace, jejíž jádro je napsáno ve skriptovacím jazyku PHP. Logicky vše začíná u souboru *index.php*, od kterého se vše dále odvíjí. Do tohoto souboru jsou postupně podle potřeby vkládány a načítány všechny další php soubory. Nalezneme zde rozcestník, který určí, jaké moduly budou nahrány pro příslušnou položku v menu systému Virtlab, tak aby se nám vždy naloadovala požadovaná stránka s korektním obsahem. Ve zmíněném rozcestníku nalezneme položku s identifikátorem VC_PAGE_SNAPSHOT, která při zavolání vloží do skriptu soubor *snapshot_management.php*, jehož úkolem je načíst stránku, kde si uživatel Virtlabu může ověřit, kolik má celkem alokováno diskového prostoru pro ukládání snapshotů a také kolik volného místa ještě uživateli zbývá. Pokud má uživatel nějaké uložené snapshoty z dřívějška, tak na této stránce je může odstranit, v případě, že například vyčerpal přidělenou diskovou kvótu a potřebuje si ukládat další snapshoty. V menu se tato stránká skrývá pod položkou *Správa snapshotů*.

Další položku v menu, která je s virtualizací spojena, se nazývá *aktivní rezervace*. Pokud ve Virtlabu žádné aktivní rezervace neprobíhají, stránka je prázdná. Pokud je však nějaká rezervace aktivní, tak se zde kromě klasického obsahu stránek pro aktivní rezervaci, který je však pro nás v tuto chvíli nezajímavý, zobrazí také část s virtuálními PC, které se v dané síťové topologii vyskytují. Stránka aktivní rezervace má přidělený identifikátor VC_PAGE_RESER_ACTIVE a pro načtení a zobrazení této stránky je použit skript *reser-active.php*. Do tohoto souboru je poté vložen soubor *snapshot-reser-active.php*, který je zodpovědný za samotnou správu virtualizovaných stanic nad aktuálně rezervovanou topologií sítě. Komunikace s virtuálním serverem, kde je nahraný nástroj XEN a kompletní správa pro virtualizaci stanic, probíhá na základě SSH protokolu, který z Virtlabu odesílá parametrizované požadavky na virtuální server, který požadavek přijme, vykoná ho a pošle zpět synchronní zprávu s odpovědí, ať už kladnou či zápornou.

Zdrojové soubory Virtlabu spojené s virtualizací

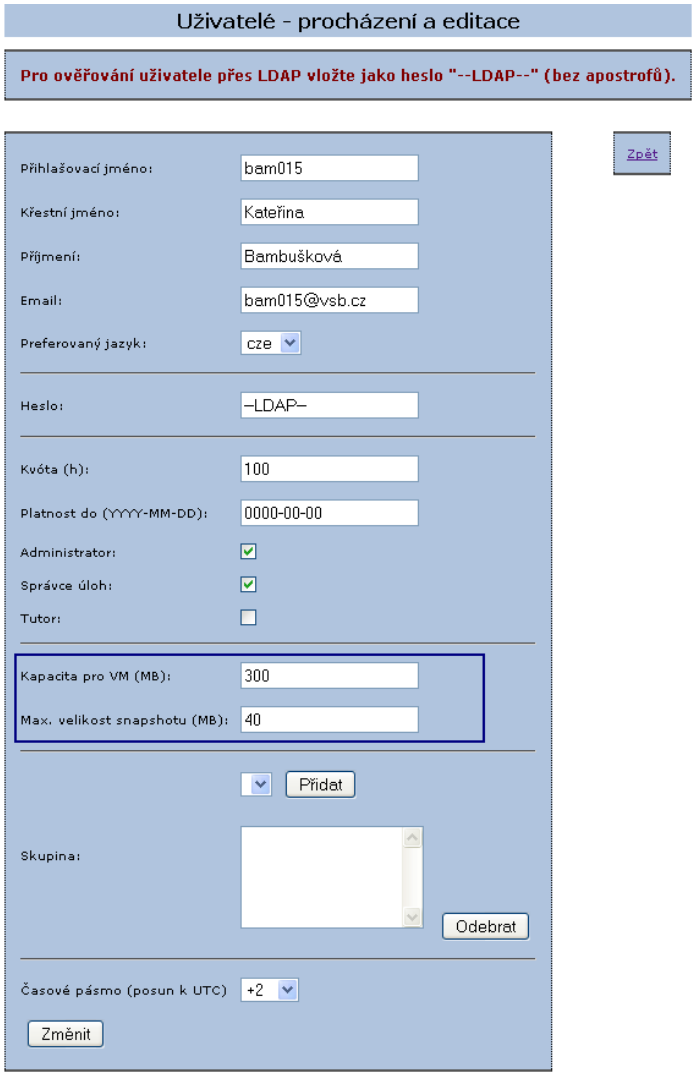
- **index.php** – hlavní soubor, do kterého jsou postupně vloženy všechny ostatní soubory
- **snapshot_management.php** – tento skript informuje uživatele o vyhrazeném diskovém prostoru pro ukládání snapshotů a také kolik volného místa ještě uživateli zbývá, uživatel zde má možnost své dříve uložené snapshoty odstranit
- **reser-active.php** – soubor zobrazující stránku s aktiní rezervací a z části také soubor obsahující funkcionalitu aktivní rezervace
- **snapshot-reser-active.php** – soubor zajišťující správů virtuálních instancí pro aktuální aktivní rezervaci
- **xen_inst_virt.sh** – skript, který odesílá parametrizované požadavky z Virtlabu na Xen server v podobě SSH spojení

5.1 Vizualizace správy virtualizovaných stanic ve Virtlabu

Posledním nutným krokem k tomu, aby studenti systému Virtlab mohli začít pracovat s virtuálními počítači na jejich síťových topologiích, je zakomponování ovládacích prvků správy virtualizovaných stanic do Virtlabu.

Správa disku pro ukládání snapshotů

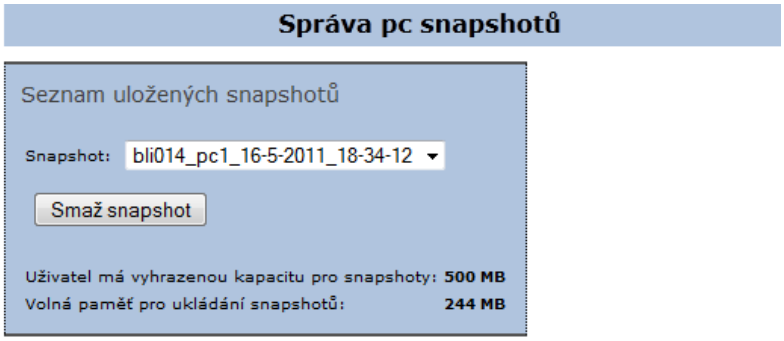
Jelikož kapacita diskového prostoru pro ukládání snapshotů virtuálních stanic není nekonečná, je administrátor Virtlabu oprávněn nastavit velikost diskového prostoru vyhrazeného pro ukládání snapshotů, který budou moci studenti využít pro vlastní použití. Administrátor zde může rovněž nastavit maximální velikost, do jaké může jeden snapshot narůst.



Obrázek 5.1: Správa disku pro ukládání snapshotů

Správa snapshotů

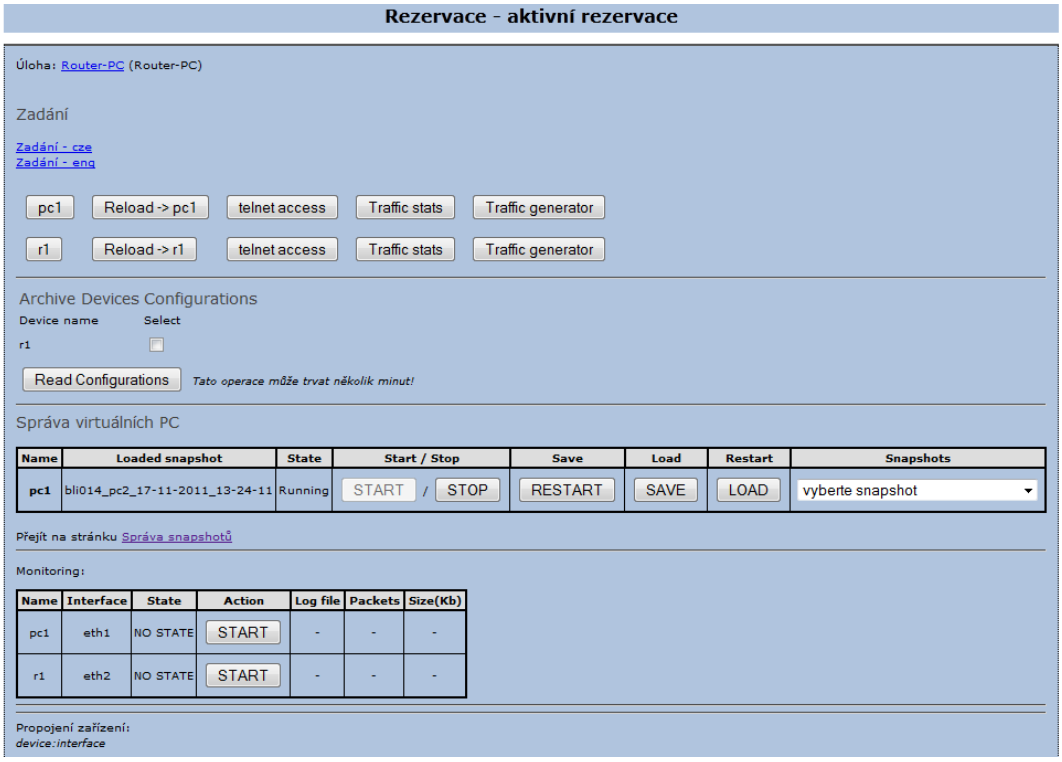
Na této stránce si uživatel můžet ověřit zbývající diskouvou kapacitu pro ukládání snapshotů a případně si zde nějaké uložené snapshoty odstranit, pokud již například uživatel vyčerpal diskovou kvótu, která je na této stránce rovněž zobrazena.



Obrázek 5.2: Správa snapshotů

Správa virtualizovaných stanic

Na stránce *aktivní rezervace* student nalezne část se správou virtuálních PC. Z tohoto místa student ovládá chod všech virtuálních PC pro danou rezervaci. Je zde vypsána informace o stavu virtuální instance (běžící/zastavená), zda byl pro daný virtuální PC nahrán snapshot nebo zda virtuální PC používá defaultní neuložený snapshot a v neposlední řadě také akce pro ovládání jednotlivýchvirtuálních instancí jako – Start, Stop, Save, Load, Restart.



Obrázek 5.3: Správa virtualizovaných stanic u aktivní rezervace

6 Závěr

Cílem této diplomové práce bylo seznámit se s architekturou školního systému Virtlab a na základě nabytých znalostí poté navrhnout a implementovat správu virtualizovaných stanic v systému Virtlab, včetně řešení archivace souborových systémů virtualizovaných stanic tak, aby studenti měli možnost zálohy virtuální instance pro její pozdější znovu užití. Posléze už pouze zbývalo takto navržené a implementované řešení zakomponovat do stávající funkcionality systému Virtlab. Systém Virtlab na základě dostupných síťových laboratorních prvků, i včetně vzdálených, vytvoří dočasnou počítačovou síť, se kterou student může libovolně pracovat a vzdělávat se tak v oblasti síťových technologií. Virtualizované stanice na takto vytvořené síti zastávají úlohu koncových počítačů, které může student libovolně konfigurovat a ověřit si zde specifika dané síťe.

Žádné konkrétní měřitelné požadavky pro vypracování diplomové práce nebyly stanoveny, ať už požadavky na použití specifické technologie či požadavky kvalitativní. I přesto se však od tohoto řešení očekává, že správa virtualizovaných stanic bude pracovat spolehlivě, s minimem údržby a také bude dostatečně rychlá. První dvě kritéria závisí především na kvalitě implementace. Naproti tomu poslední kritérium – rychlost a odezva závisí hlavně na použitých technologiích. Věřím, že první dva předpoklady tato diplomová práce splňuje a třetí předpoklad naopak splňuje virtualizační platforma XEN, která díky technologii zvané paravirtulizace dosahuje ztrátu výkonu pouze v řádu jednotek procent ve srovnání s klasickými systémy, které běží přímo na fyzickém hardwaru.

Tato diplomová práce studentům umožňuje, aby na síti vytvořené Virtlabem, mohli s virtuálními počítači pracovat bez omezení, jako by to byly skutečné počítače a rovněž si také rozpracovanou virtuální instanci uložit za účelem znovu použití na jiných sítích, aby student pokaždé nemusel vše konfigurovat od začátku. Z předcházejícího vyplívá, že požadavky na funkcionalitu správy virtualizovaných stanic byly splněny, i přesto tato práce nabízí celou řadu dalších rozšíření. Další vývoj projektu by se tedy mohl ubírat směrem rozšíření stávající funkcionality a zapracování na detailech, které by studentům práci s virtualizovanými instancemi zpříjemnily a zefektivnily. Například by bylo možné spustit virtuální stroj s již přednastaveným síťovým prostředím na základě studentových požadavků, které by předvyplnil do interaktivního formuláře. Velice užitečné rozšíření funkcionality by zajisté byla možnost logování síťových statistik u virtuálních strojů s následnou přehlednou prezentací nasbíraných dat.

Virtualizace je v dnešním IT světě fenomén, který v době krize pomáhá firmám šetřit nemalé náklady za štíhlejší IT infrastrukturu a tím i potažmo energie a prostory. Během svého vývoje našla virtualizace uplatnění i v mnoha dalších oblastech a oborech, díky čemuž se vizualizace stala zcela běžnou a masově rozšířenou technologií a na základě dosavadního vývoje virtualizace její vliv do budoucna ještě určitě vzroste.

7 Literatura

[1] VON HAGEN, William. *Professional Xen Virtualization*. Indiana: Wrox, 2008. ISBN 978-0470138113.

[2] JUŠKA, Pavel. *Virtualizacní platforma pro dynamickou aktivaci specifických virtuálních instancí s OS Linux*. Ostrava, 2010. 49 s. Diplomová práce. VŠB-TU Ostrava, FEI.

[3] *Stránky projektu Virtlab* [online]. 2006-06-28 [cit. 2010-11-05]. Virtuální laboratoř počítačových sítí. Dostupné z: http://www.virtlab.cz

[4] *Xen Hypervisor* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://www.xen.org/

[5] *Xen Documentation* [online]. 2005-2010 [cit. 2012-04-25]. Dostupné z: http://www.xen.org/support/documentation.html

[6] *Xen Architecture* [online]. 29.06.2010 [cit. 2012-04-25]. Dostupné z: http://wiki.xensource.com/xenwiki/XenArchitecture

[7] *Xen Networking* [online]. 29.09.2011 [cit. 2012-04-25]. Dostupné z: http://wiki.xensource.com/xenwiki/XenNetworking

[8] Virtualizace. *Wikipedia* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://cs.wikipedia.org/wiki/Virtualizace

[9] Paravirtualizace. *MUNI ICS* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://www.ics.muni.cz/bulletin/articles/545.html

[10] Virtualizace s podporou hardwaru. *Vratislav Podzimek* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://www.fi.muni.cz/~kas/p090/referaty/2011-jaro/ut/virt.html

[11] Emulátor. *Wikipedia* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://cs.wikipedia.org/wiki/Emul%C3%A1tor

[12] Hypervizor. *Wikipedia* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://cs.wikipedia.org/wiki/Hypervizor

[13] DIRBÁK, Ivan: *Xen – základy virtualizácie* [online], 19.11.2008, [cit. 2010-4-03] Dostupné z: http://www.abclinuxu.cz/clanky/system/xen-zaklady-virtualizacie.

[14] BOUREK, Jiří: *Xen vs. KVM – Souboj v extrémních podmínkách* [online], 6. 5. 2010, [cit. 2010-4-03] Dostupné z: http://www.abclinuxu.cz/clanky/xen-vs.-kvm-souboj-v-extremnich-podminkach.

[15] *Logical Volume Management (LVM)* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://sourceware.org/lvm2/

[16] Bash. *Wikipedia* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://en.wikipedia.org/wiki/Bash_%28Unix_shell%29

[17] Debian Packages. *Debian* [online]. 2012 [cit. 2012-04-25]. Dostupné z: http://www.debian.org/distrib/packages

Přílohy

Obsah CD

Adresář	Popis
/system/aplikace	implementační skripty pro virtualizaci Virtlabu
/system/config	konfigurační soubory potřebné pro Xen
/text	text diplomové práce